

Fig. 1

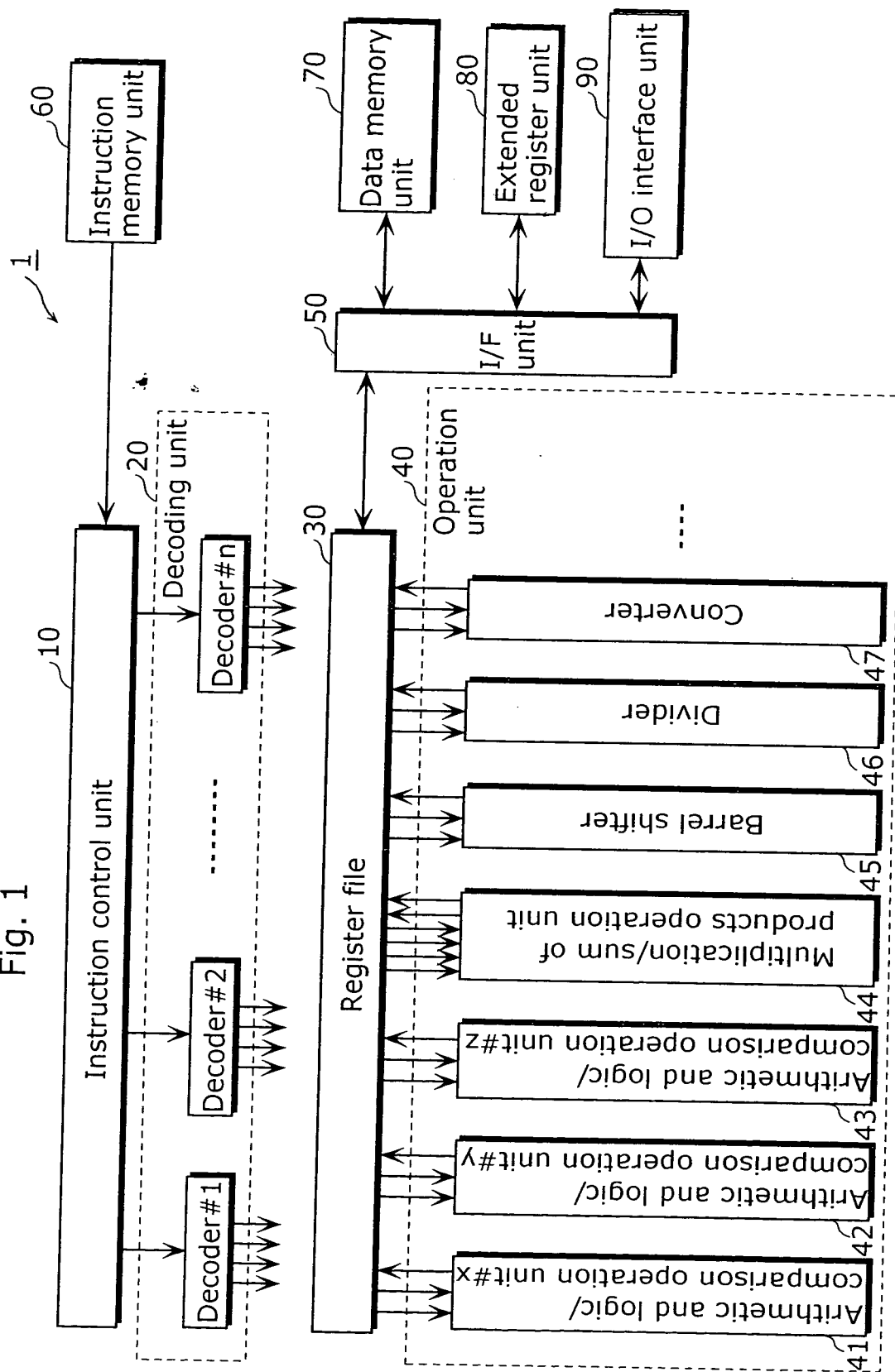


Fig. 2

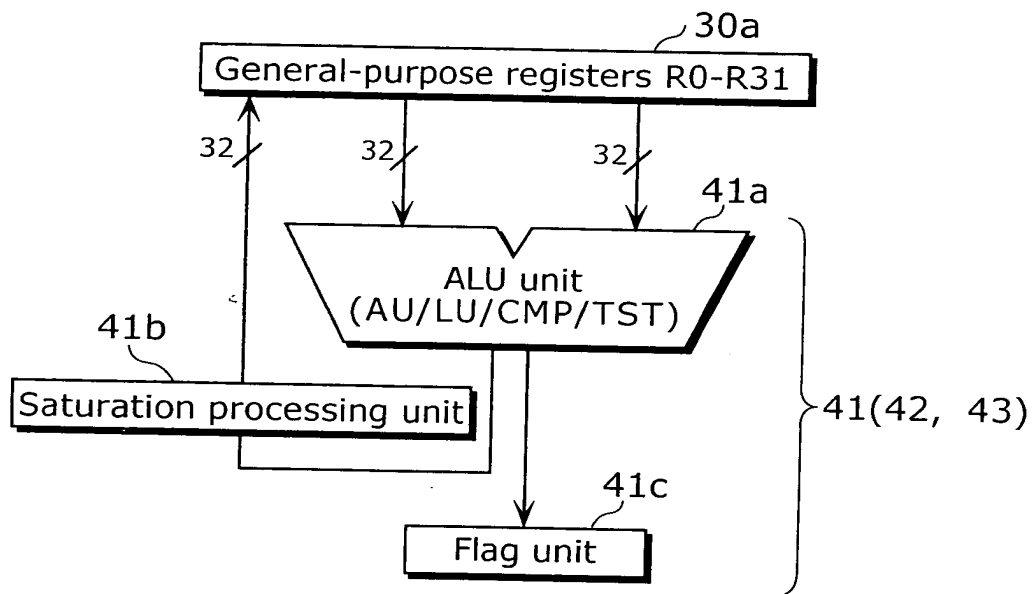


Fig. 3

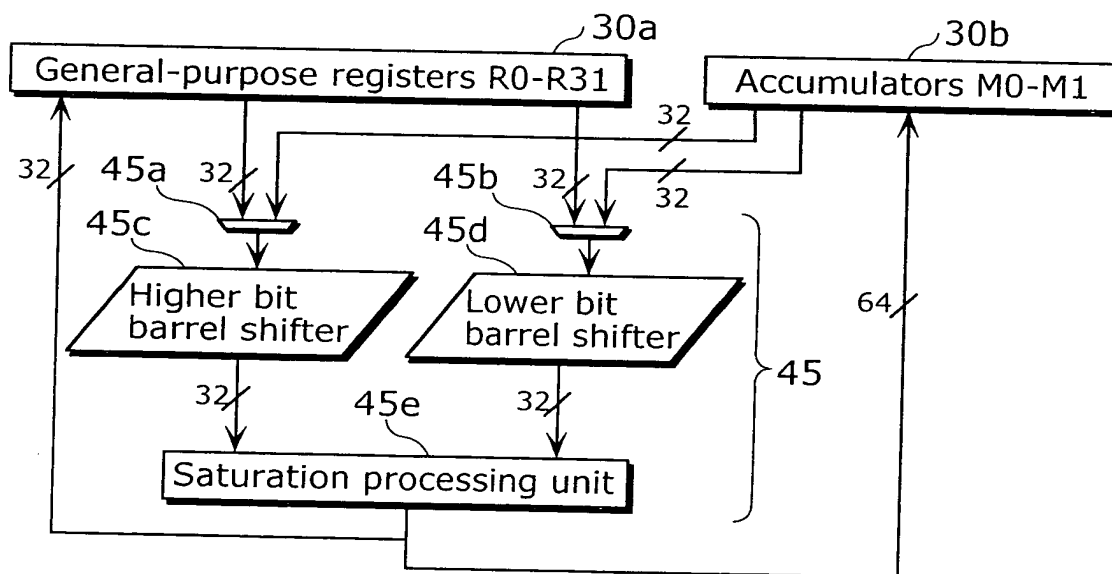


Fig. 4

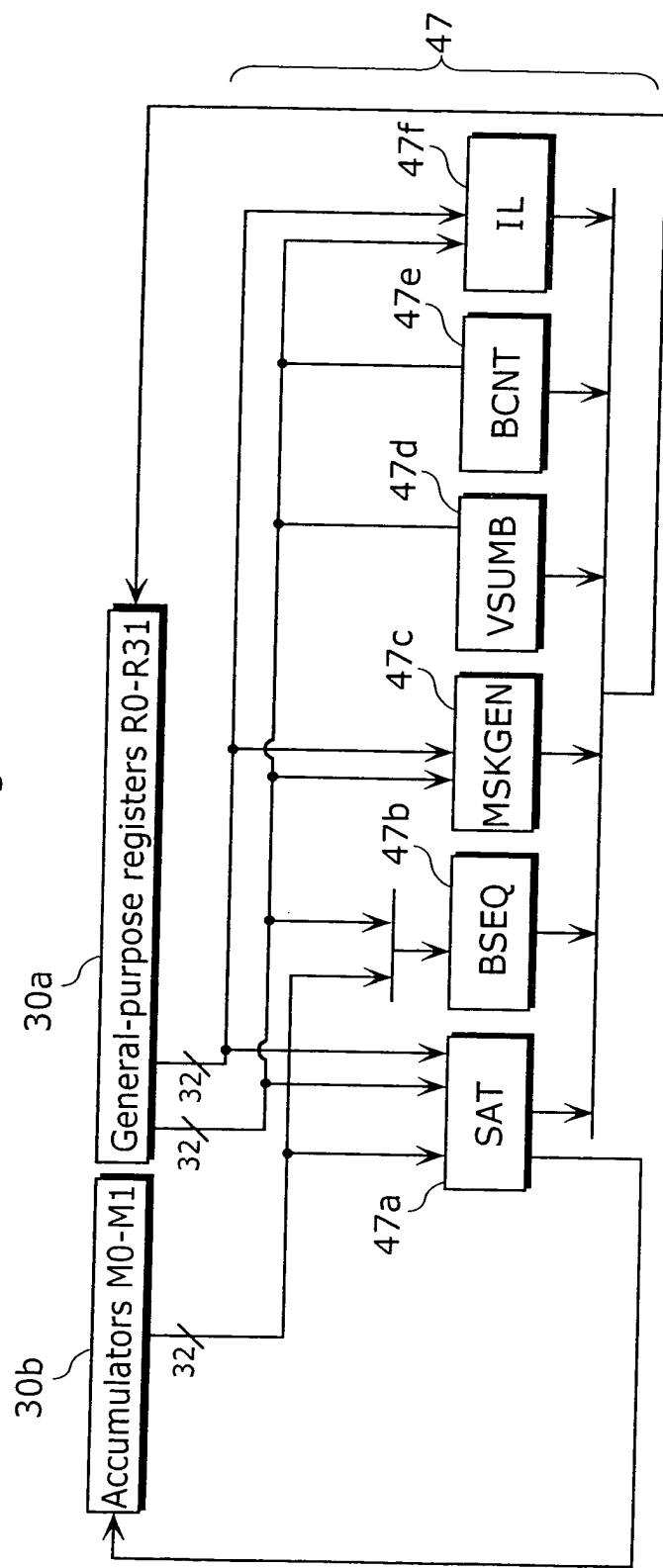


Fig. 5

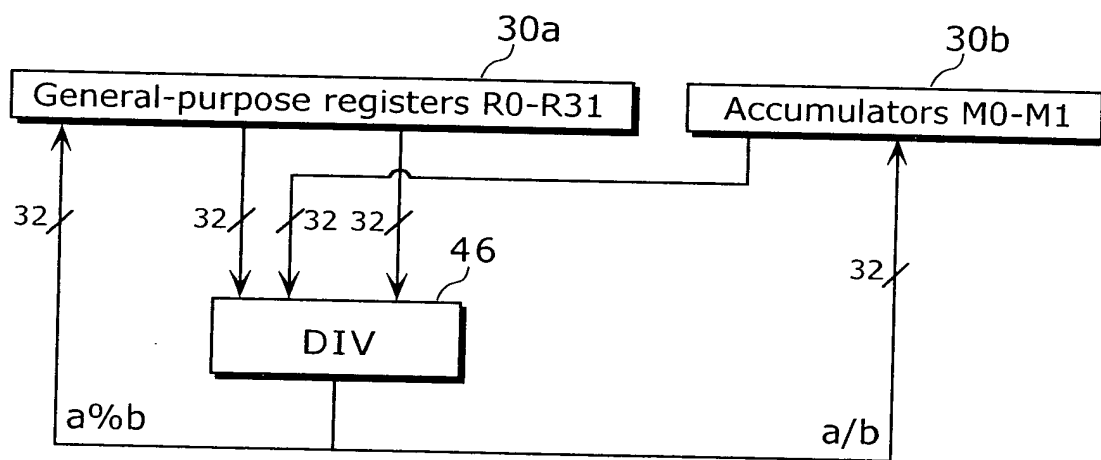


Fig. 6

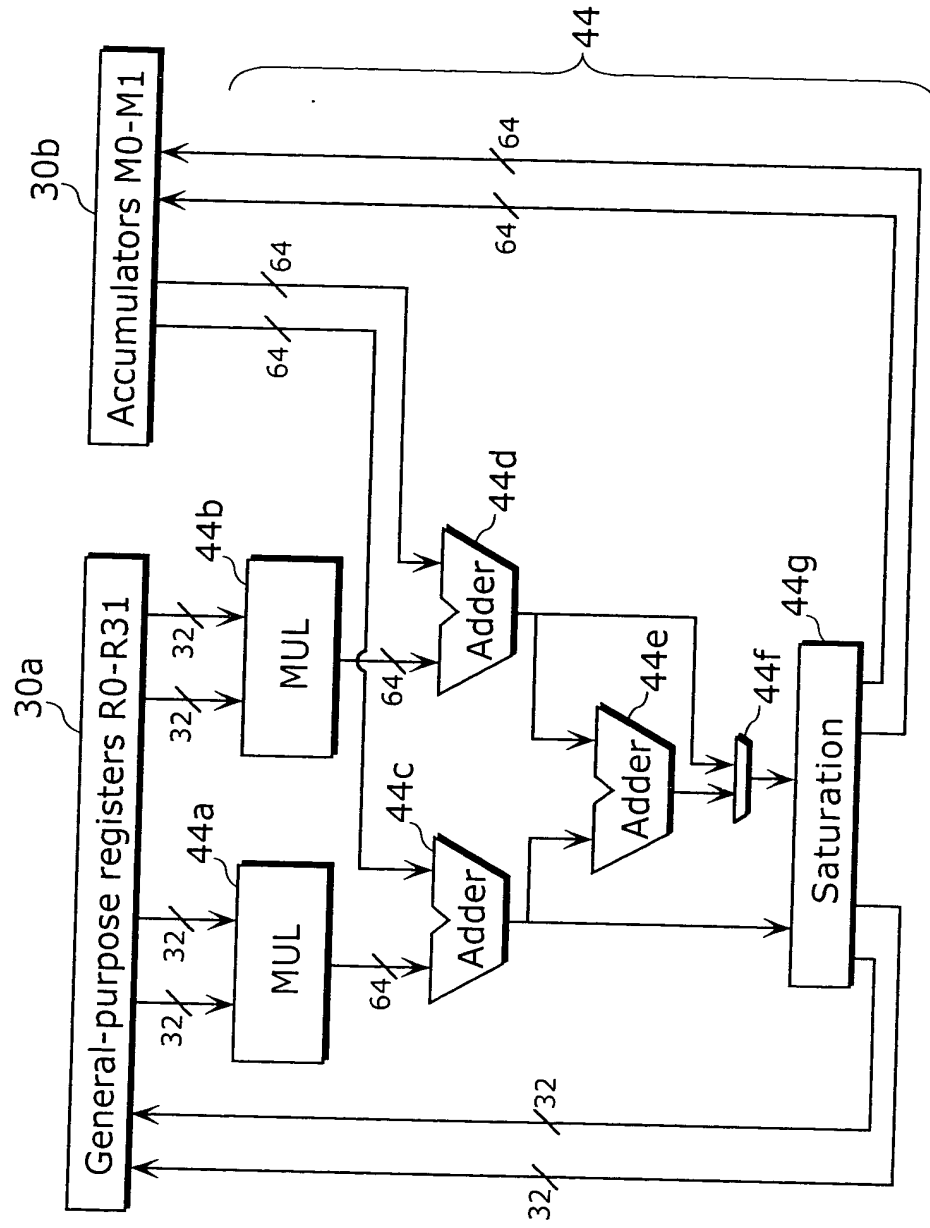


Fig. 7

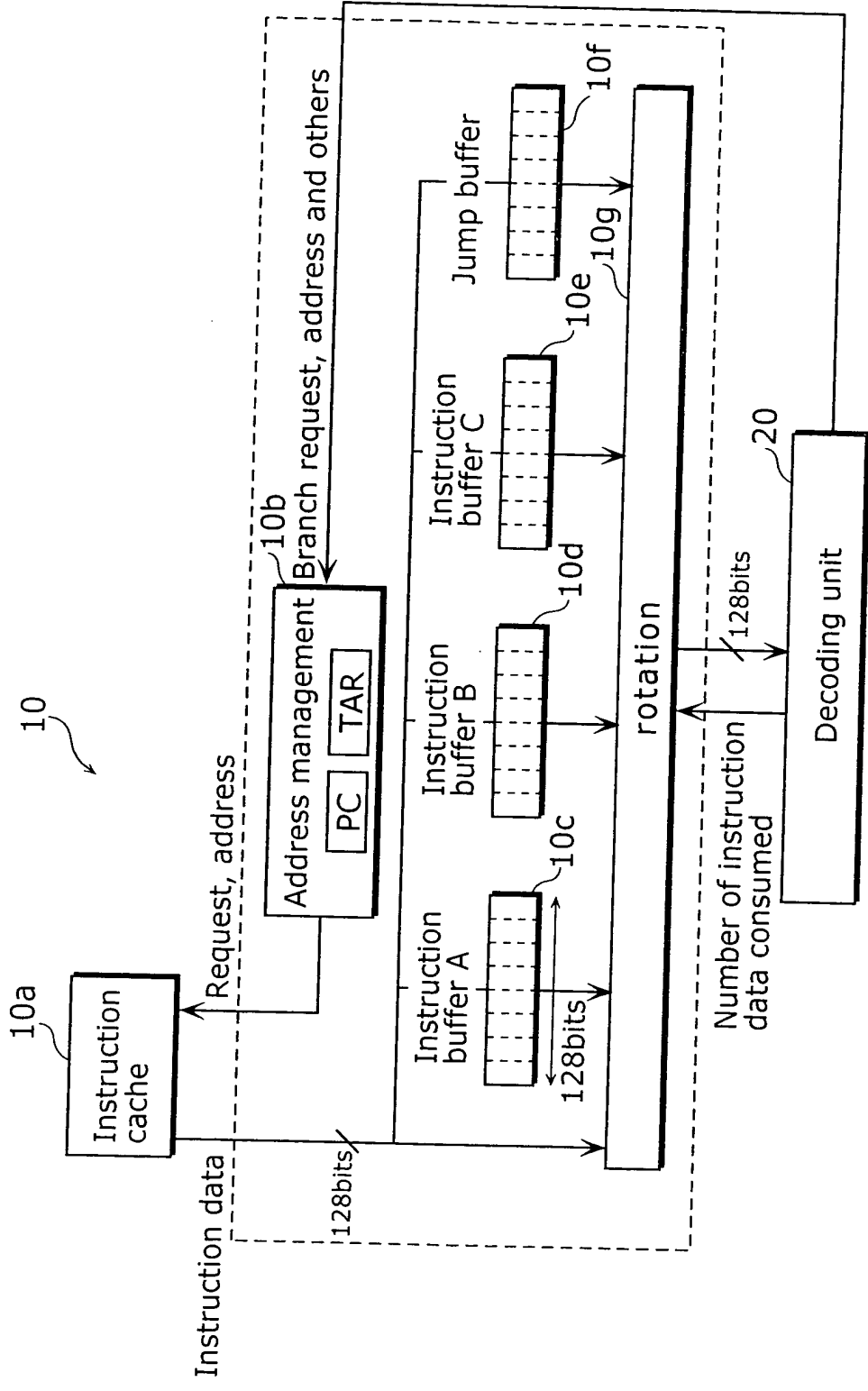


Fig. 8

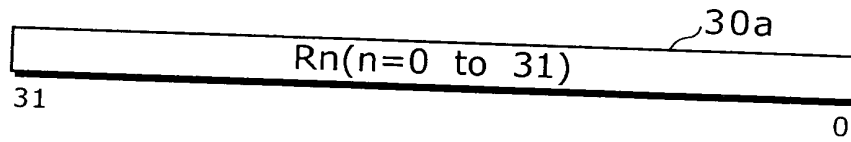


Fig. 9

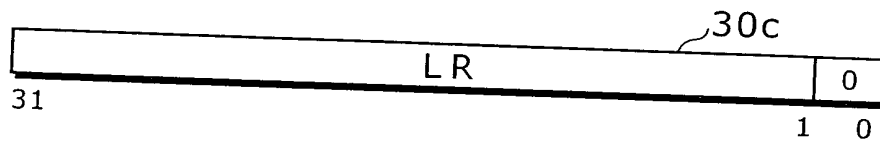


Fig. 10



Fig. 11

Bit	31	30	29	28	27	26	25	24
Bit name	reserved	SWE	FXP	reserved	IH	EH	PL	
Bit	23	22	21	20	19	18	17	16
Bit name	FIE3	FIE2	FIE1	FIE0	reserved	reserved	AEE	IE
Bit	15	14	13	12	11	10	9	8
Bit name	Reserved							
Bit	7	6	5	4	3	2	1	0
Bit name	IM[7:0]							

Fig. 12

32

Bit	31	30	29	28	27	26	25	24
Bit name	ALN		reserved	BPO				
Bit	23	22	21	20	19	18	17	16
Bit name	reserved			VC3		VC2	VC1	VC0
Bit	15	14	13	12	11	10	9	8
Bit name	reserved						OVS	CAS
Bit	7	6	5	4	3	2	1	0
Bit name	C7	C6	C5	C4	C3	C2	C1	C0

Fig. 13A

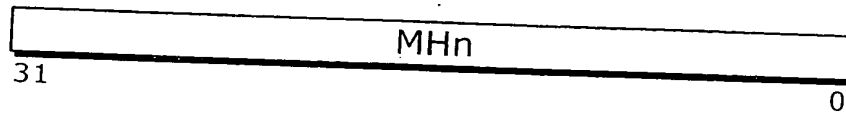
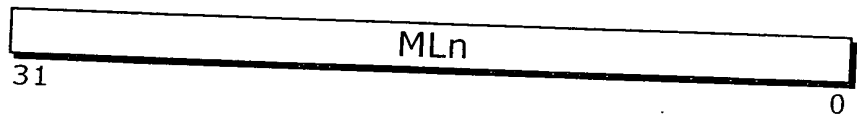


Fig. 13B



30b

Fig. 14

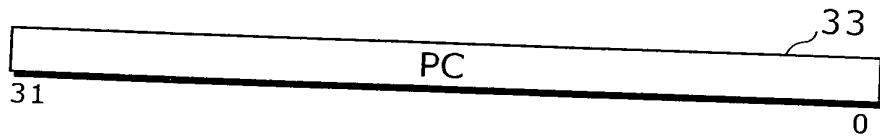


Fig. 15

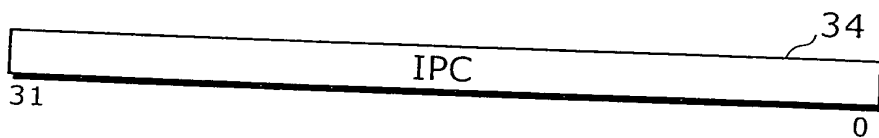


Fig. 16

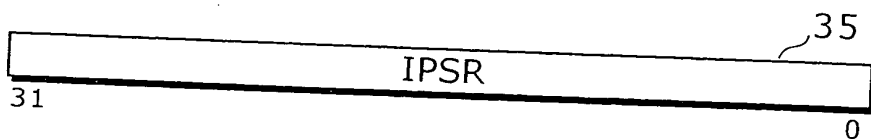


Fig. 17

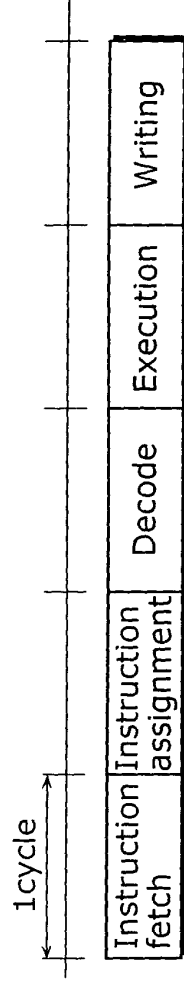


Fig. 18

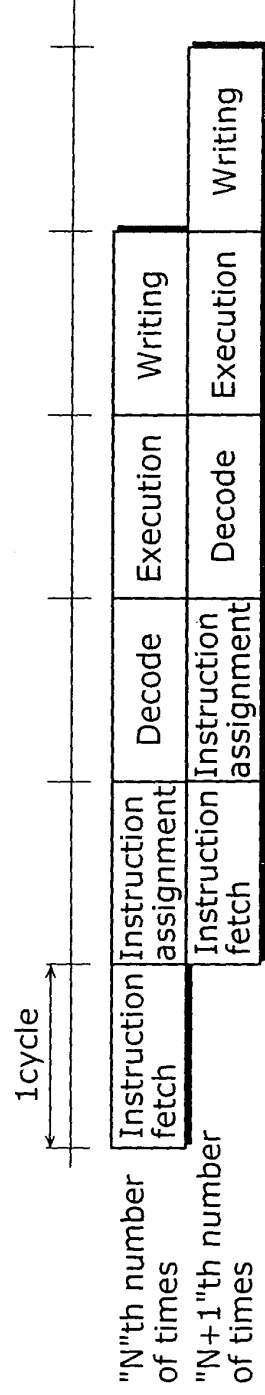


Fig. 19

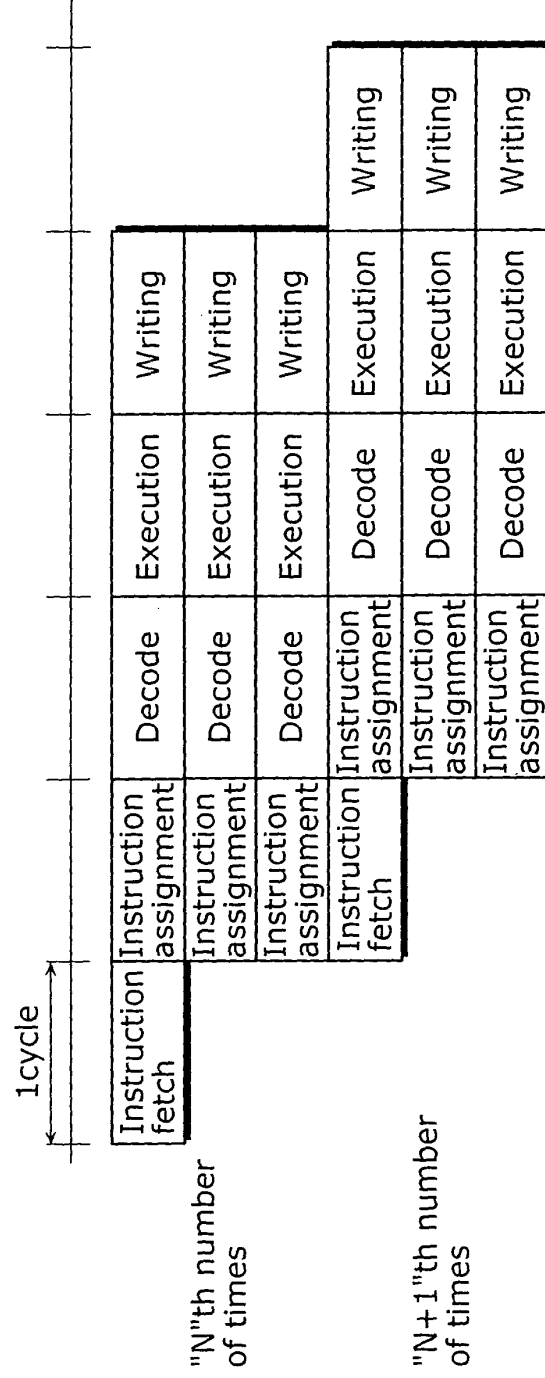


Fig. 20

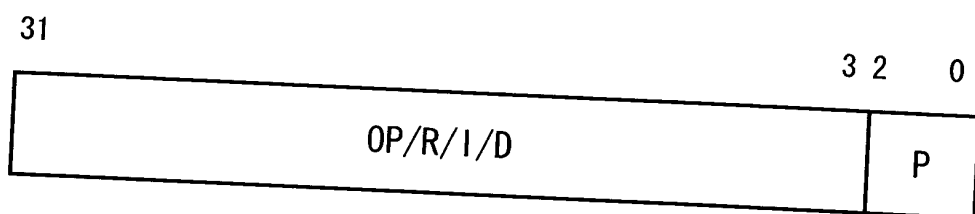


Fig. 21

Category	SIMD	Size	Instruction	Operand	CFR	PSR	Typical behavior	Operation unit	31 16
ALU add system	S I M D	Word	add	Rc,Ra,Rb Rb,Ra,i12s SP,i19s Ra2,Rb2 Rc3,Ra3,Rb3 Ra2,i05s SP,i11s					32
			addu	Rb,GP,i16u Rb,SP,i16u Ra3,SP,i08u					16
			addc	Rc,Ra,Rb	Wcas,c0:c1		Addition with carry		32
			addvw	Rc,Ra,Rb	Wovs		Addition with overflow		16
			adds	Rc,Ra,Rb			$Ra + Rb \rightarrow Rb \gg 1$		32
			addsr	Rc,Ra,Rb			$Ra + Rb + 1 \rightarrow Rb \gg 1$		32
			s1add	Rc,Ra,Rb Rc3,Ra3,Rb3			$Ra + Rb \rightarrow Rc$		16
			s2add	Rc,Ra,Rb Rc3,Ra3,Rb3			$Ra + Rb (>2) \rightarrow Rc$		32
			addmsk	Rc,Ra,Rb	RtBP0		$Ra \oplus Rb \rightarrow Rc$		16
			addarvw	Rc,Ra,Rb					16
		Half word	faddvh	Rc,Ra,Rb	Wovs				32
		S I M D	vaddh	Rc,Ra,Rb			$Ra + Rb \rightarrow Rb$		32
			vaddhvh	Rc,Ra,Rb	Wovs		$Ra + Rb \rightarrow Rb$		32
			vsaddh	Rb,Ra,i08s			$Ra + Rb \rightarrow Rb$		32
			vaddsh	Rc,Ra,Rb			$Ra + Rb \rightarrow Rb$		32
			vaddsrh	Rc,Ra,Rb			$Ra + Rb (+1) \rightarrow Rb$		32
			vaddhvc	Rc,Ra,Rb	RVC				32
			vaddrhvc	Rc,Ra,Rb					32
			vxaddh	Rc,Ra,Rb			$Ra + Rb \rightarrow Rb$		32
			vxaddhvh	Rc,Ra,Rb	Wovs		$Ra + Rb \rightarrow Rb$		32
			vhaddh	Rc,Ra,Rb			$Ra + Rb \rightarrow Rb$		32
			vhaddhvh	Rc,Ra,Rb	Wovs		$Ra + Rb \rightarrow Rb$		32
			vladdh	Rc,Ra,Rb			$Ra + Rb \rightarrow Rb$		32
			vladdhvh	Rc,Ra,Rb	Wovs		$Ra + Rb \rightarrow Rb$		32
		Byte	vaddb	Rc,Ra,Rb			$Ra + Rb \rightarrow Rb$		32
			vsaddb	Rb,Ra,i08s			$Ra + Rb \rightarrow Rb$		32
			vaddsb	Rc,Ra,Rb			$Ra + Rb \rightarrow Rb$		32
			vaddsrb	Rc,Ra,Rb			$Ra + Rb (+1)(+1)(+1) \rightarrow Rb$		32

Fig. 22

Category	SIMD	Size	Instruction	Operand	CFR	PSR	Typical behavior	Operation unit	
ALU sub system	S I N G L E	Word	sub	Rc,Rb,Ra Rb2,Ra2 Rc3,Rb3,Ra3					31
			rsub	Rb,Ra,i08s Ra2,Rb2 Ra2,i04s			Immediate value - \boxed{Rs} \rightarrow \boxed{Rb} ($\boxed{Rb2}$)	32	
			subc	Rc,Rb,Ra	W:cas,c0c1		With carry		
			subvw	Rc,Rb,Ra	W:ovs		With overflow		
			subvs	Rc,Rb,Ra			$\boxed{Ra} - \boxed{Rb} \rightarrow \boxed{Rc}$		
			submsk	Rc,Rb,Ra	R:BP0		\boxed{Ra} $\boxed{CF,BP0}$ \rightarrow \boxed{Rc} \boxed{Rb}		
	S I M D	Half word	fsubvh	Rc,Rb,Ra					
			vsubh	Rc,Rb,Ra			\boxed{Ra} $\boxed{10}$ $\boxed{16}$ \boxed{Rb} $\boxed{10}$ $\boxed{16}$ \rightarrow $\boxed{16}$ $\boxed{16}$ \boxed{Rc}		
			vsubhvh	Rc,Rb,Ra	W:ovs				
			vsrsubh	Rb,Ra,i08s			Immediate value - \boxed{Ra} $\boxed{16}$ $\boxed{16}$ \boxed{Rb} $\boxed{16}$ $\boxed{16}$ \rightarrow $\boxed{16}$ $\boxed{16}$ \boxed{Rb}		
			vsubsh	Rc,Rb,Ra			\boxed{Ra} $\boxed{10}$ $\boxed{16}$ \boxed{Rb} $\boxed{10}$ $\boxed{16}$ \rightarrow $\gg 1$ $\gg 1$ $\boxed{16}$ $\boxed{16}$ \boxed{Rc}		
		Half word	vsubsh	Rc,Rb,Ra			\boxed{Ra} $\boxed{10}$ $\boxed{16}$ \boxed{Rb} $\boxed{10}$ $\boxed{16}$ \rightarrow $\gg 1$ $\gg 1$ $\boxed{16}$ $\boxed{16}$ \boxed{Rc}		
			vxsubh	Rc,Rb,Ra			\boxed{Ra} $\boxed{10}$ $\boxed{16}$ \boxed{Rb} $\boxed{10}$ $\boxed{16}$ \rightarrow $\boxed{16}$ $\boxed{16}$ \boxed{Rc}		
			vxsubhvh	Rc,Rb,Ra	W:ovs				
			vhsbsh	Rc,Rb,Ra			\boxed{Ra} $\boxed{10}$ $\boxed{16}$ \boxed{Rb} $\boxed{10}$ $\boxed{16}$ \rightarrow $\boxed{16}$ $\boxed{16}$ \boxed{Rc}		
			vhsbshvh	Rc,Rb,Ra	W:ovs				
Byte	vsubb	Rc,Rb,Ra			\boxed{Ra} $\boxed{16}$ $\boxed{16}$ \boxed{Rb} $\boxed{16}$ $\boxed{16}$ \rightarrow $\boxed{16}$ $\boxed{16}$ \boxed{Rc}				
	vsrsubb	Rb,Ra,i08s			(Immediate value) \boxed{Ra} $\boxed{8}$ $\boxed{8}$ $\boxed{8}$ $\boxed{8}$ \boxed{Rb} $\boxed{8}$ $\boxed{8}$ $\boxed{8}$ $\boxed{8}$ \rightarrow $\boxed{8}$ $\boxed{8}$ $\boxed{8}$ $\boxed{8}$ \boxed{Rc}				
	vasubh	Rc,Rb,Ra	R:V/C						

Fig. 23

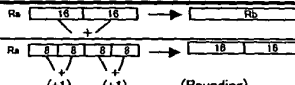
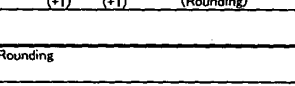
Category	SIMD	Size	Instruction	Operand	CFR	PSR	Typical behavior	Operation unit	31 16
ALU logic system	SINGLE	Word	and	Rc,Ra,Rb Rb,Ra,i08u Ra2,Rb2			AND	A	32
			andn	Rc,Ra,Rb Rb,Ra,i08u Ra2,Rb2					16
			or	Rc,Ra,Rb Rb,Ra,i08u Ra2,Rb2			OR		32
			xor	Rc,Ra,Rb Rb,Ra,i08u Ra2,Rb2			Exclusive OR		16
									32
ALU mov system	SINGLE	Word	mov	Rb,Reg32 Reg32,Rb Rb2,Reg16 Reg16,Rb2 Ra2,Rb Ra,i16s Ra2,i08s			Reg32 = TAR LR SVR PSR CFR MH0 MH1 ML0 ML1 EPSR IPC IPSR PC EPC PSR0 PSR1 PSR2 PSR3 CFR0 CFR1 CFR2 CFR3 Reg16 = TAR LR MH0 MH1	A	32
			movp	Rc,Rc+1,Ra,Rb			Rc ← -Ra; Rc+1 ← -Rb;		16
			movcf	Ck,Cj,Cm,Cn			Ci ← -Cj; Cm ← -Cn;		32
			mvclvcs	Cm:Cm+1	W:ovs		Cm:Cm+1 ← -CFR.OVS; CFR.OVS ← -0;		16
			mvclcas	Cm:Cm+1	W:cas		Cm:Cm+1 ← -CFR.CAS; CFR.CAS ← -0;		32
			sethi	Ra,i16s					16
									32
ALU max min system	SINGLE	Word	max	Rc,Ra,Rb	W:c0:c1		Rc ← max(Ra,Rb)	A	32
		Half word	min	Rc,Ra,Rb	W:c0:c1		Rc ← min(Ra,Rb)		
		Byte	vmaxh	Rc,Ra,Rb					
			vminh	Rc,Ra,Rb					
			vmaxb	Rc,Ra,Rb					
ALU abs system	SINGLE	Half word	abs	Rb,Ra			Absolute value	A	32
			absvw	Rb,Ra	W:ovs		With overflow		
			fabsvh	Rb,Ra	W:ovs				
			vabshvh	Rb,Ra	W:ovs				
ALU neg system	SINGLE	Word	negvw	Rb,Ra	W:ovs			A	32
		Half word	fnegvh	Rb,Ra	W:ovs				
		Byte	vnegvh	Rb,Ra	W:ovs				
ALU sum system	SIMD	Half word	vsumh	Rb,Ra				A	32
			vsumh2	Rb,Ra					
			vsumrh2	Rb,Ra					
			vabssumb	Rc,Ra,Rb					
ALU o t h e r	SINGLE	SI MD	frndvh	Rb,Ra	W:ovs		Rounding	C	32
			vfrndvh	Rb,Mn	W:ovs				
			vsel	Rc,Ra,Rb	RVC				
			vsgnh	Rb,Ra					

Fig. 24

Category	SIMD	Size	Instruction	Operand	CFR	PSR	Typical behavior	Operation unit	31 16
CMP	S I N G L E		cmpCCn	Cm,Ra,Rb,Cn Cm,Ra,i05s,Cn Cm:Cm+1,Ra,Rb,Cn Cm:Cm+1,Ra,i05s,Cn	W:CF		CC = eq, ne, gt, ge, gtu, geu, le, lt, leu, leu Cm <- result & Cn; (Cm+1 <- ~result & Cn;)	A	31 16
			cmpCCa	Cm:Cm+1,Ra,Rb,Cn Cm:Cm+1,Ra,i05s,Cn	W:CF		Cm <- result & Cn; Cm+1 <- ~(result & Cn);		32
			cmpCCo	Cm:Cm+1,Ra,Rb,Cn Cm:Cm+1,Ra,i05s,Cn	W:CF		Cm <- result Cn; Cm+1 <- ~(result Cn);		16
			cmpCC	C6,Ra2,Rb2 C6,Ra2,i04s	W:CF		CC = eq, ne, gt, ge, le, lt C6 <- result		16
			tstzn	Cm,Ra,Rb,Cn Cm,Ra,i05u,Cn Cm:Cm+1,Ra,Rb,Cn Cm:Cm+1,Ra,i05u,Cn	W:CF		Cm <- (Ra & Rb == 0) & Cn; (Cm+1 <- ~(Ra & Rb == 0) & Cn;)		32
			tstza	Cm:Cm+1,Ra,Rb,Cn Cm:Cm+1,Ra,i05u,Cn	W:CF		Cm <- (Ra & Rb == 0) & Cn; Cm+1 <- ~((Ra & Rb == 0) & Cn);		32
			tstzo	Cm:Cm+1,Ra,Rb,Cn Cm:Cm+1,Ra,i05u,Cn	W:CF		Cm <- (Ra & Rb == 0) Cn; Cm+1 <- ~((Ra & Rb == 0) Cn);		16
			tstnn	Cm,Ra,Rb,Cn Cm,Ra,i05u,Cn Cm:Cm+1,Ra,Rb,Cn Cm:Cm+1,Ra,i05u,Cn	W:CF		Cm <- (Ra & Rb != 0) & Cn; (Cm+1 <- ~(Ra & Rb != 0) & Cn;)		32
			tstna	Cm:Cm+1,Ra,Rb,Cn Cm:Cm+1,Ra,i05u,Cn	W:CF		Cm <- (Ra & Rb != 0) & Cn; Cm+1 <- ~((Ra & Rb != 0) & Cn);		16
			tstno	Cm:Cm+1,Ra,Rb,Cn Cm:Cm+1,Ra,i05u,Cn	W:CF		Cm <- (Ra & Rb != 0) Cn; Cm+1 <- ~((Ra & Rb != 0) Cn);		32
			tstz	C6,Ra2,Rb2 C6,Ra2,i04u	W:CF		C6 <- (Ra2 & Rb2 == 0)		16
			tstn	C6,Ra2,Rb2 C6,Ra2,i04u	W:CF		C6 <- (Ra2 & Rb2 != 0)		32
	S I M D	Half word	vcmpCCh	Ra,Rb					
		Byte	vscmpCCh	Ra,i05s	W:CF		CC = eq, ne, gt, le, ge, lt		
	S I M D	Half word	vcmpCCb	Ra,Rb					
		Byte	vscmpCCb	Ra,i05s	W:CF		CC = eq, ne, gt, le, ge, lt		

Fig. 25

Category	SIMD	Size	Instruction	Operand	CFR	PSR	Typical behavior	Operation unit	31 16
mul system	S I N G L E	Word x Word	mul	Mm,Rc,Ra,Rb Mm,Rb,Ra,i08s				X2	
			mulu	Mm,Rc,Ra,Rb Mm,Rb,Ra,i08s			Unsigned multiplication		
			fmulww	Mm,Rc,Ra,Rb		fxp	Fixed point operation		
		Word x Half word	hmul	Mm,Rc,Ra,Rb				X1	
			lmul	Mm,Rc,Ra,Rb					
			fmulhw	Mm,Rc,Ra,Rb		fxp			
		Half word x Half word	fmulhw	Mm,Rc,Ra,Rb		fxp		X1	
			fmulhh	Mm,Rc,Ra,Rb		fxp			
			fmulhr	Mm,Rc,Ra,Rb		fxp			
			fmulhr	Mm,Rc,Ra,Rb		fxp			
			vmul	Mm,Rc,Ra,Rb					
			vfmulw	Mm,Rc,Ra,Rb		fxp			
		H A L F W O R D x H A L F W O R D	vfmulh	Mm,Rc,Ra,Rb		fxp		X2	32
			vfmulhr	Mm,Rc,Ra,Rb		fxp			
			vvmul	Mm,Rc,Ra,Rb					
			vxfmulw	Mm,Rc,Ra,Rb		fxp			
			vxfmulh	Mm,Rc,Ra,Rb		fxp			
			vxfmulhr	Mm,Rc,Ra,Rb		fxp			
			vfmul	Mm,Rc,Ra,Rb					
			vfmulw	Mm,Rc,Ra,Rb		fxp			
			vfmulh	Mm,Rc,Ra,Rb		fxp			
			vfmulhr	Mm,Rc,Ra,Rb		fxp			
			vpfmulww	Mm,Rc,Rc+1,Ra,Rb Mm,Rc,Rc+1,Ra,Rb		fxp	With rounding		

Fig. 26

Category	SIMD	Size	Instruction	Operand	CFR	PSR	Typical behavior	Operation unit	31 16
mac system	S I N G L E	Word × Word	mac	Mn,Rc,Ra,Rb,Mn M0,Rc,Ra,Rb,Rx			Sum of products operation using mul	X2	31 16
			fmacww	Mn,Rc,Ra,Rb,Mn M0,Rc,Ra,Rb,Rx		fxp	Sum of products operation using fmulww		32
		Word × Half word	hmac	Mn,Rc,Ra,Rb,Mn M0,Rc,Ra,Rb,Rx			Sum of products operation using hmul	X1	16
			lmac	Mn,Rc,Ra,Rb,Mn M0,Rc,Ra,Rb,Rx			Sum of products operation using lmul		32
			fmacww	Mn,Rc,Ra,Rb,Mn M0,Rc,Ra,Rb,Rx		fxp	Sum of products operation using fmulhww		16
		Half word × Half word	fmacww	Mn,Rc,Ra,Rb,Mn M0,Rc,Ra,Rb,Rx		fxp	Sum of products operation using fmulhw		32
			fmacww	Mn,Rc,Ra,Rb,Mn M0,Rc,Ra,Rb,Rx		fxp	Sum of products operation using fmulhw		
			fmacww	Mn,Rc,Ra,Rb,Mn M0,Rc,Ra,Rb,Rx		fxp	Sum of products operation using fmulhw		
			fmacww	Mn,Rc,Ra,Rb,Mn M0,Rc,Ra,Rb,Rx		fxp	Sum of products operation using fmulhw		
	S I M D	H A L F W O R D	vmac	Mn,Rc,Ra,Rb,Mn M0,Rc,Ra,Rb,Rx			Sum of products operation using vmul	X2	16
			vfmaw	Mn,Rc,Ra,Rb,Mn M0,Rc,Ra,Rb,Rx		fxp	Sum of products operation using vfmulw		32
			vfmaw	Mn,Rc,Ra,Rb,Mn M0,Rc,Ra,Rb,Rx		fxp	Sum of products operation using vfmulw		16
			vfmaw	Mn,Rc,Ra,Rb,Mn M0,Rc,Ra,Rb,Rx		fxp	Sum of products operation using vfmulw		
			vfmaw	Mn,Rc,Ra,Rb,Mn M0,Rc,Ra,Rb,Rx		fxp	Sum of products operation using vfmulw		
			vfmaw	Mn,Rc,Ra,Rb,Mn M0,Rc,Ra,Rb,Rx		fxp	Sum of products operation using vfmulw		
		H A L F W O R D	vhmac	Mn,Rc,Ra,Rb,Mn M0,Rc,Ra,Rb,Rx			Sum of products operation using vhmul		
			vfmaw	Mn,Rc,Ra,Rb,Mn M0,Rc,Ra,Rb,Rx		fxp	Sum of products operation using vhmulw		
			vhmac	Mn,Rc,Ra,Rb,Mn M0,Rc,Ra,Rb,Rx		fxp	Sum of products operation using vhmulh		
			vhmac	Mn,Rc,Ra,Rb,Mn M0,Rc,Ra,Rb,Rx		fxp	Sum of products operation using vhmulh		
			vhmac	Mn,Rc,Ra,Rb,Mn M0,Rc,Ra,Rb,Rx		fxp	Sum of products operation using vhmulh		
			vhmac	Mn,Rc,Ra,Rb,Mn M0,Rc,Ra,Rb,Rx		fxp	Sum of products operation using vhmulh		
		W O R D	vmac	Mn,Rc,Ra,Rb,Mn M0,Rc,Ra,Rb,Rx			Sum of products operation using vmul		32
			vfmaw	Mn,Rc,Ra,Rb,Mn M0,Rc,Ra,Rb,Rx		fxp	Sum of products operation using vfmulw		
			vfmaw	Mn,Rc,Ra,Rb,Mn M0,Rc,Ra,Rb,Rx		fxp	Sum of products operation using vfmulw		
			vfmaw	Mn,Rc,Ra,Rb,Mn M0,Rc,Ra,Rb,Rx		fxp	Sum of products operation using vfmulw		
			vfmaw	Mn,Rc,Ra,Rb,Mn M0,Rc,Ra,Rb,Rx		fxp	Sum of products operation using vfmulw		
			vfmaw	Mn,Rc,Ra,Rb,Mn M0,Rc,Ra,Rb,Rx		fxp	Sum of products operation using vfmulw		
		Word × Half word	vfmaw	Mn,Rc,Ra,Rb,Mn M0,Rc,Ra,Rb,Rx		fxp	Sum of products operation using vfmulw		

Fig. 27

Category	SIMD	Size	Instruction	Operand	CFR	PSR	Typical behavior	Operation unit	31 16
msu system	S I N G L E	Word x Word	msu	Mm,Rc,Ra,Rb,Mn M0,Rc,Ra,Rb,Rx			Difference of products operation using mul	X2	
			fmsuww	Mm,Rc,Ra,Rb,Mn M0,Rc,Ra,Rb,Rx		fxp	Difference of products operation using fmulww		
		Word x Half word	hmsu	Mm,Rc,Ra,Rb,Mn M0,Rc,Ra,Rb,Rx			Difference of products operation using hmul	X1	
			lmsu	Mm,Rc,Ra,Rb,Mn M0,Rc,Ra,Rb,Rx			Difference of products operation using lmul		
			fmsuhww	Mm,Rc,Ra,Rb,Mn M0,Rc,Ra,Rb,Rx		fxp	Difference of products operation using fmulhww		
			fmsuhw	Mm,Rc,Ra,Rb,Mn M0,Rc,Ra,Rb,Rx		fxp	Difference of products operation using fmulhw		
		Half word x Half word	fmsuhh	Mm,Rc,Ra,Rb,Mn M0,Rc,Ra,Rb,Rx		fxp	Difference of products operation using fmulhh		
			fmsuhhr	Mm,Rc,Ra,Rb,Mn M0,Rc,Ra,Rb,Rx		fxp	With rounding		
		H A L F W O R D x H A L F W O R D	vmsu	Mm,Rc,Ra,Rb,Mn M0,Rc,Ra,Rb,Rx			Difference of products operation using vmul	X2	
			vfmsuw	Mm,Rc,Ra,Rb,Mn		fxp	Difference of products operation using vmul		
	vfmsuh		Mm,Rc,Ra,Rb,Mn M0,Rc,Ra,Rb,Rx		fxp	Difference of products operation using vmulh			
	vxmsu		Mm,Rc,Ra,Rb,Mn M0,Rc,Ra,Rb,Rx			Difference of products operation using vxmul			
	vxfmsuw		Mm,Rc,Ra,Rb,Mn		fxp	Difference of products operation using vxfmulw			
	vxfmsuh		Mm,Rc,Ra,Rb,Mn M0,Rc,Ra,Rb,Rx		fxp	Difference of products operation using vxfmulh			
	vhmsu		Mm,Rc,Ra,Rb,Mn M0,Rc,Ra,Rb,Rx			Difference of products operation using vhmul			
	vhfmsuw		Mm,Rc,Ra,Rb,Mn		fxp	Difference of products operation using vhfmulw			
	vhfmsuh		Mm,Rc,Ra,Rb,Mn M0,Rc,Ra,Rb,Rx		fxp	Difference of products operation using vhfmulh			
	vlmsu		Mm,Rc,Ra,Rb,Mn M0,Rc,Ra,Rb,Rx			Difference of products operation using vlmul			
	vlfmsuw	Mm,Rc,Ra,Rb,Mn		fxp	Difference of products operation using vlfmulw				
	vlfmsuh	Mm,Rc,Ra,Rb,Mn M0,Rc,Ra,Rb,Rx		fxp	Difference of products operation using vlfmulh				

Fig. 28

Category	SIMD	Size	Instruction	Operand	CFR	PSR	Typical behavior	Operation unit	31 16
MEM id system	S I M D	Word	ld	Rb,(Ra,d10u) Rb,(GP,d13u) Rb,(SP,d13u) Rb,(Ra+);10s Rb2,(Ra2) Rb2,(Ra2,d05u) Rb2,(GP,d06u) Rb2,(SP,d06u) Rb2,(Ra2+)			Register 32 ← Memory 32	M	32
			ldh	Rb,(Ra,d09u) Rb,(GP,d12u) Rb,(SP,d12u) Rb,(Ra+);09s Rb2,(Ra2) Rb3,(Ra3,d04u) Rb2,(GP,d05u) Rb2,(SP,d05u) Rb2,(Ra2+)			32 ← 16		16
			ldhu	Rb,(Ra,d09u) Rb,(GP,d12u) Rb,(SP,d12u) Rb,(Ra+);09s					32
			ldb	Rb,(Ra,d08u) Rb,(GP,d11u) Rb,(SP,d11u) Rb,(Ra+);08s			Register 32 ← Memory 8		16
		Byte	ldbu	Rb,(Ra,d08u) Rb,(GP,d11u) Rb,(SP,d11u) Rb,(Ra+);08s					32
			ldbh	Rb,(Ra+);07s			16 16 ← 8 8		
		Half word	ldbuh	Rb,(Ra+);07s			32 32 ← 32 32		
			ldp	Rb:Rb+1,(Ra,d11u) LR:SVR,(Ra,d11u) TAR:UDR,(Ra,d11u) Rb:Rb+1,(GP,d14u) LR:SVR,(GP,d14u) TAR:UDR,(GP,d14u) Rb:Rb+1,(SP,d14u) LR:SVR,(SP,d14u) TAR:UDR,(SP,d14u) Rb:Rb+1,(Ra+);11s Rb:Rb+1,(SP,d07u) LR:SVR,(SP,d07u) Rb2:Re2,(Ra2+)					16
		Word	ldhp	Rb:Rb+1,(Ra,d10u) Rb:Rb+1,(Ra+);10s Rb2:Re2,(Ra2+)			32 32 ← 16 16		32
			ldbp	Rb:Rb+1,(Ra,d09u) Rb:Rb+1,(Ra+);09s			32 32 ← 8 8		16
		Byte	ldbhp	Rb:Rb+1,(Ra+);07s			16 16 ← 8 8 8 8		
			ldbuhp	Rb:Rb+1,(Ra+);07s					32
		Half word	ldbhp	Rb:Rb+1,(Ra+);07s					
			ldbuhp	Rb:Rb+1,(Ra+);07s					

Fig. 29

Category	SIMD	Size	Instruction	Operand	CFR	PSR	Typical behavior	Operation unit		
MEM store system	S I M D	Word	st	(Ra,d10u),Rb (GP,d13u),Rb (SP,d13u),Rb (Ra+),i10s,Rb (Ra2),Rb2 (Ra2,d05u),Rb2 (GP,d06u),Rb2 (SP,d06u),Rb2 (Ra2+),Rb2			Register 32 → Memory 32	M	31 16	
									32	
									16	
									32	
		Half word	sth	(Ra,d09u),Rb (GP,d12u),Rb (SP,d12u),Rb (Ra+),i09s,Rb (Ra2),Rb2 (Ra2,d04u),Rb2 (GP,d05u),Rb2 (SP,d05u),Rb2 (Ra2+),Rb2			16 → 16		32	
								16		
								32		
								16		
		Byte	stb	(Ra,d08u),Rb (GP,d11u),Rb (SP,d11u),Rb (Ra+),i08s,Rb			8 → 8		32	
		Byte→ Half word	stbh	(Ra+),i07s,Rb			8 8 → 16		32	
		P A I R	Word	stp	(Ra,d11u),Rb:Rb+1 (Ra,d11u),LR:SVR (Ra,d11u),TAR:UDR (GP,d14u),Rb:Rb+1 (GP,d14u),LR:SVR (GP,d14u),TAR:UDR (SP,d14u),Rb:Rb+1 (SP,d14u),LR:SVR (SP,d14u),TAR:UDR (Ra+),i11s,Rb:Rb+1 (SP,d07u),Rb:Re (SP,d07u),LR:SVR (Ra2+),Rb2:Re2			32 32 → 32 32		32
									16	
									32	
									16	
			Half word	sthp	(Ra,d10u),Rb:Rb+1 (Ra+),i10s,Rb:Rb+1 (Ra2+),Rb2:Re2			16 16 → 32		16
			Byte	stbp	(Ra,d09u),Rb:Rb+1 (Ra+),i09s,Rb:Rb+1			8 8 → 16		16
			Byte→ Half word	stbhp	(Ra+),i07s,Rb:Rb+1			8 8 8 8 → 32		32

Fig. 30

Category	SIMD	Size	Instruction	Operand	CFR	PSR	Typical behavior	Operation unit	
BRA			setlr	d09s C5,d09s			Set LR Store instruction fetched from LR in branch buffer	B	31 16
			settar	d09s C6,C2:C4,d09s C6,Cm,d09s C6,C4,d09s	W:c6 W:c2:c4,c6 W:c6,cm W:c6		Set TAR Store instruction fetched from TAR in branch buffer		32
			setbb	LR TAR			Store instruction fetched from LR in branch buffer Store instruction fetched from TAE in branch buffer		16
			jloop	C5,LR,Ra,i08s C6,TAR,Ra,i08s C6,C2:C4,TAR,Ra,i08s C6,Cm,TAR,Ra,i08s C6,TAR,Ra2 C6,C2:C4,TAR,Ra2 C6,Cm,TAR,Ra2	W:c5 W:c6 W:c2:c4,c6 W:c6,cm W:c6 W:c2:c4,c6 W:c6		Only predicate [c5] Only predicate [c6]		32
			jmp	TAR LR					16
			jmp1	TAR LR	R:CF				
			jmpf	TAR LR Cm,TAR C6,C2:C4,TAR					
			jmprr	LR					
			br	d20s d09s				32 16 32 16	
			brl	d20s d09s	R:CF		Only predicates [c6][c7]		
			rti			W:PSR R:eh			

Fig. 31

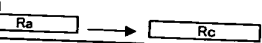
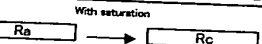
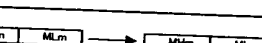

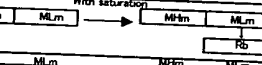
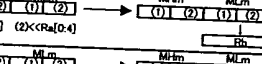
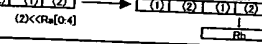
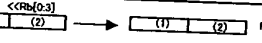
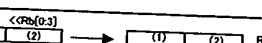
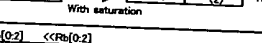

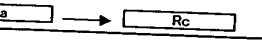
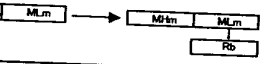
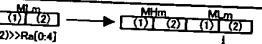
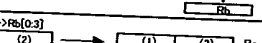
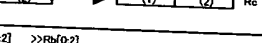
Category	SIMD	Size	Instruction	Operand	CFR	PSR	Typical behavior	Operation unit	31 16
BS asl system	S I N G L E	Word	asl	Rc,Ra,Rb Rb,Ra,i05u Ra2,i04u			Left shift $\ll(Ra[0:4])$ 	S1	32 16
			faslvw	Rc,Ra,Rb Rb,Ra,i05u Rc,Ra,Rb Rb,Ra,i05u	W:ovs		With saturation 		
		Pair word	aslp	Mm,Ra,Mn,Rb Mm,Rb,Mn,i06u Mm,Rc,MHn,Ra,Rb Mm,Rb,MHn,Ra,i06u			$\ll(Ra[0:4])$ 	S2	32
			faslpvw	Mm,Ra,Mn,Rb Mm,Rb,Mn,i06u	W:ovs		With saturation 		
	S I M D	Word	vasl	Mm,Ra,Mn,Rb Mm,Rb,Mn,i05u				S2	32
			vfaslvw	Mm,Ra,Mn,Rb Mm,Rb,Mn,i05u	W:ovs				
		Half word	vaslh	Rc,Ra,Rb Rb,Ra,i04u			$\ll(Rb[0:3])$ $\ll(Rb[0:3])$ 	S1	32
			vfaslvh	Rc,Ra,Rb Rb,Ra,i04u	W:ovs		With saturation 		
		Byte	vaslb	Rc,Ra,Rb Rb,Ra,i03u			$\ll(Rb[0:2])$ $\ll(Rb[0:2])$ 	S1	32
									
BS asr system	S I N G L E	Word	asr	Rc,Ra,Rb Rb,Ra,i05u Ra2,i04u			Arithmetic shift right $\gg(Rb[0:4])$ 	S1	32 16
		Pair word	asrp	Mm,Ra,Mn,Rb Mm,Rb,Mn,i06u Mm,Rc,MHn,Ra,Rb Mm,Rb,MHn,Ra,i06u			$\gg(Ra[0:4])$ 	S2	32
	S I M D	Word	vasr	Mm,Ra,Mn,Rb Mm,Rb,Mn,i05u				S2	32
		Half word	vasrh	Rc,Ra,Rb Rb,Ra,i04u			$\gg(Rb[0:3])$ $\gg(Rb[0:3])$ 		
		Byte	vasrb	Rc,Ra,Rb Rb,Ra,i03u			$\gg(Rb[0:2])$ $\gg(Rb[0:2])$ 	S1	32
									

Fig. 32


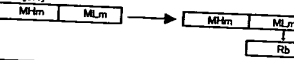
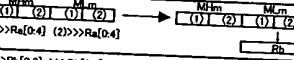
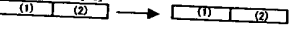
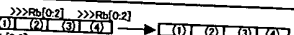
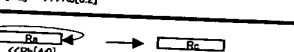
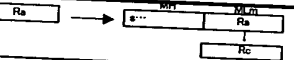
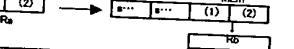
Category	SIMD	Size	Instruction	Operand	CFR	PSR	Typical behavior	Operation unit	31 16	
BS lsr system	SINGLE	Word	lsr	Rc,Ra,Rb Rb,Ra,i05u			Logical shift right >>>Ra[0:4] 	S1	32	
		Pair word	lsrp	Mm,Ra,Mn,Rb Mm,Rb,Mn,i06u Mm,Rc,MHn,Ra,Rb Mm,Rb,MHn,Ra,i06u			>>>Ra[0:4] >>>Ra[0:4] 	S2		
	SIMD	Word	vlsr	Mm,Ra,Mn,Rb Mm,Rb,Mn,i05u			Mm Mn (1) (2) (1) (2) >>>Ra[0:4] >>>Ra[0:4] 	S2		
		Half word	vlshr	Rc,Ra,Rb Rb,Ra,i04u			>>>Ra[0:3] >>>Ra[0:3] 	S1		
		Byte	vlrb	Rc,Ra,Rb Rb,Ra,i03u			>>>Ra[0:2] >>>Ra[0:2] 			
BS rotate system	SINGLE	Word	rol	Rc,Ra,Rb Rb,Ra,i05u				S1	32	
	SIMD	Half word	vrolh	Rc,Ra,Rb Rb,Ra,i04u						
		Byte	vrolb	Rc,Ra,Rb Rb,Ra,i03u						
BS ext system	SINGLE	Word	extw	Mm,Rb,Ra				C	32	
		Half word	exth	Ra2			Ra2 (1) >>>Ra2 (1) Ra2	S2		
			exthu	Ra2			Ra2 (1) >>>Ra2 (1) Ra2			
		Byte	extb	Ra2			Ra2 (1) >>>Ra2 (1) Ra2			
			extbu	Ra2			Ra2 (1) >>>Ra2 (1) Ra2			
	SIMD	Half word	vexth	Mm,Rb,Ra			(1) (2) >>>Ra2 (1) (2) Ra2 	C	32	

Fig. 33

Category	SIMD	Size	Instruction	Operand	CFR	PSR	Typical behavior	Operation unit	31 16
CNV valn system	SIMD		valn	Rc,Ra,Rb	Rc[n[1:0]		$\ll \text{GERALN}[1:0] \ll 3$	C	32
			valn1	Rc,Ra,Rb			$\ll \text{GERALN}[1:0] \ll 3$		
			valn2	Rc,Ra,Rb			$\ll \text{GERALN}[1:0] \ll 3$		
			valn3	Rc,Ra,Rb			$\ll \text{GERALN}[1:0] \ll 3$		
			valnvc1	Rc,Ra,Rb	RVC0		$\ll \text{GERALN}[1:0] \ll 3$		
			valnvc2	Rc,Ra,Rb	RVC0				
			valnvc3	Rc,Ra,Rb	RVC0				
			valnvc4	Rc,Ra,Rb	RVC0				

Fig. 34

Category	SIMD	Size	Instruction	Operand	CFR	PSR	Typical behavior	Operation unit	31 16
CNV	S I N G L E		bcntl	Rb,Ra			Count the number of 1s	C	32
			bseq0	Rb,Ra			Count number of values from MSB until first 0 is reached		
			bseq1	Rb,Ra			Count number of values from MSB until first 1 is reached		
			bseq	Rb,Ra			Count number of values from MSB until first -1 is reached		
			mskbrvh	Rc,Ra,Rb	RBP0				
			byterev	Rb,Ra					
			mskbrvb	Rc,Ra,Rb	RBP0				
	S I M D	Half word	vintlh	Rc,Ra,Rb					
			vintlh	Rc,Ra,Rb					
		Byte	vintlb	Rc,Ra,Rb					
			vintlb	Rc,Ra,Rb					
		Half word	vhunpkh	Rb:Rb+1,Ra					
			vhunpkh	Rb:Rb+1,Ra					
		Byte	vhunpkb	Rb:Rb+1,Ra					
			vhunpkb	Rb:Rb+1,Ra					
		Half word	vlunpkh	Rb:Rb+1,Ra					
			vlunpkh	Rb:Rb+1,Ra					
		Byte	vlunpkb	Rb:Rb+1,Ra					
			vlunpkb	Rb:Rb+1,Ra					
		Half word	vunpk1	Rb,Mn					
			vunpk2	Rb,Mn					
		Byte	vstovh	Rb,Ra					
			vstovb	Rb,Ra					
		Byte	vhpkb	Rc,Ra,Rb					
			vhpkb	Rc,Ra,Rb					

Fig. 35

Category	SIMD	Size	Instruction	Operand	CFR	PSR	Typical behavior	Operation unit	31 16
SAT vlpk system	S I M D	Word→ Half word	vlpkh	Rc,Ra,Rb			<div> <div> <div>Ra</div> <div>(1)</div> </div> <div> <div>Rb</div> <div>(2)</div> </div> <div>With saturation</div> <div>→</div> <div> <div>(1)</div> <div>(2)</div> </div> <div>Rc</div> </div>	C	32
			vlpkhu	Rc,Ra,Rb			<div> <div> <div>Ra</div> <div>(1)</div> </div> <div> <div>Rb</div> <div>(2)</div> </div> <div>With unsigned saturation</div> <div>→</div> <div> <div>(1)</div> <div>(2)</div> </div> <div>Rc</div> </div>		
		Half word→ Byte	vlpkb	Rc,Ra,Rb			<div> <div> <div>Ra</div> <div>(1)</div> <div>(2)</div> </div> <div> <div>Rb</div> <div>(3)</div> <div>(4)</div> </div> <div>With saturation</div> <div>→</div> <div> <div>(1)</div> <div>(2)</div> <div>(3)</div> <div>(4)</div> </div> <div>Rc</div> </div>		
			vlpkbu	Rc,Ra,Rb			<div> <div> <div>Ra</div> <div>(1)</div> <div>(2)</div> </div> <div> <div>Rb</div> <div>(3)</div> <div>(4)</div> </div> <div>With unsigned saturation</div> <div>→</div> <div> <div>(1)</div> <div>(2)</div> <div>(3)</div> <div>(4)</div> </div> <div>Rc</div> </div>		
SAT sat system	S I N G L E	Word	satw	Mn,Rb,Mn			Word saturation	C	32
			sath	Rb,Ra			Half wordsaturation		
			satb	Rb,Ra			Bytesaturation		
			satbu	Rb,Ra			Unsigned byte saturation		
			sat9	Rb,Ra			9-bit saturation		
			sat12	Rb,Ra			12-bit saturation		
	S I M D	Half word	vsath	Mn,Rb,Mn			<div> <div> <div>Mn</div> <div>(0)</div> <div>(2)</div> <div>(0)</div> <div>(2)</div> </div> <div> <div>Mn</div> <div>(0)</div> <div>(2)</div> </div> <div>With saturation</div> <div>→</div> <div> <div>Mn</div> <div>(0)</div> <div>(2)</div> </div> <div>Mn</div> </div>		
			vsath8	Rb,Ra			Signed 8-bit saturation		
			vsath8u	Rb,Ra			Unsigned 8-bit saturation		
			vsath9	Rb,Ra			9-bit saturation		
			vsath12	Rb,Ra			12-bit saturation		

Fig. 36

Category	SIMD	Size	Instruction	Operand	CFR	PSR	Typical behavior	Operation unit	31
MSK			mshgen	Rc,Rb Rb,i05U,i05u			Generate mask Rb[12:8] Rb[4:0] or Rb[4:0] Rb[12:8] 0... 1... 0... Rb 1... 0... 1... Rb[12:8] Rb[4:0] Rb[4:0] Rb[12:8] 0... 1... 0... Rb 1... 0... 1...	S2	32
			msh	Rc,Ra,Rb Rb,Ra,i05U,i05u			Rb[12:8] Rb[4:0] Rb[4:0] Rb[12:8] 0... 1... 0... Rb 1... 0... 1... Rb[12:8] Rb[4:0] Rb[4:0] Rb[12:8] 0... 1... 0... Rb 1... 0... 1...		
EXTR			extr	Rc,Ra,Rb Rb,Ra,i05U,i05u			Rb[12:8] Rb[4:0] With sign extension Ra 0... 1... 0... Rb 1... 0... 1... Rb[12:8] Rb[4:0] Rb[4:0] Rb[12:8] 0... 1... 0... Rb 1... 0... 1...	S2	32
			extru	Rc,Ra,Rb Rb,Ra,i05U,i05u			Rb[12:8] Rb[4:0] (Without sign extension) Ra 0... 1... 0... Rb 1... 0... 1... Rb[12:8] Rb[4:0] Rb[4:0] Rb[12:8] 0... 1... 0... Rb 1... 0... 1...		
DIV			div divu	MHm,Rc,MHn,Ra,Rb MHm,Rc,MHn,Ra,Rb	W:ovs		Division	DIV	32
ETC			piNl			W:ih,ie,fi,pl R:PSR	Software interrupt N=0~7	B	32
			piN			W:ih,ie,pl R:PSR	Software interrupt N=0~7		16
			scN			W:ih,ie,pl R:PSR	System call N=0~7		16
			ldstb	Rb,(Ra)			load bus lock	M	32
			rd	Rb,(Ra) Rb,(d11u) Rb2,(Ra2)		R:eee	External register read		16
			wt	(Ra),Rb (d11u),Rb (Ra2),Rb2		R:eee	External register write		32
			dpref	(Ra,d11u)			Pre-fetch	DBGM	16
			dbgmn	i18u			N=0~3		16
			vcchk		W:CF R:VC		VC flag check	B	32
			vmpsw				VMP switching		32
			vmpsw	LR					32
			vmpintd1			W:ie	VMP switching disabled		32
			vmpintd2			W:ie			32
			vmpintd3			W:ie			32
			vmpinte1			W:ie	VMP switching enabled	A	32
			vmpinte2						32
			vmpinte3						32
			nop				no operation	A	16

Fig. 37

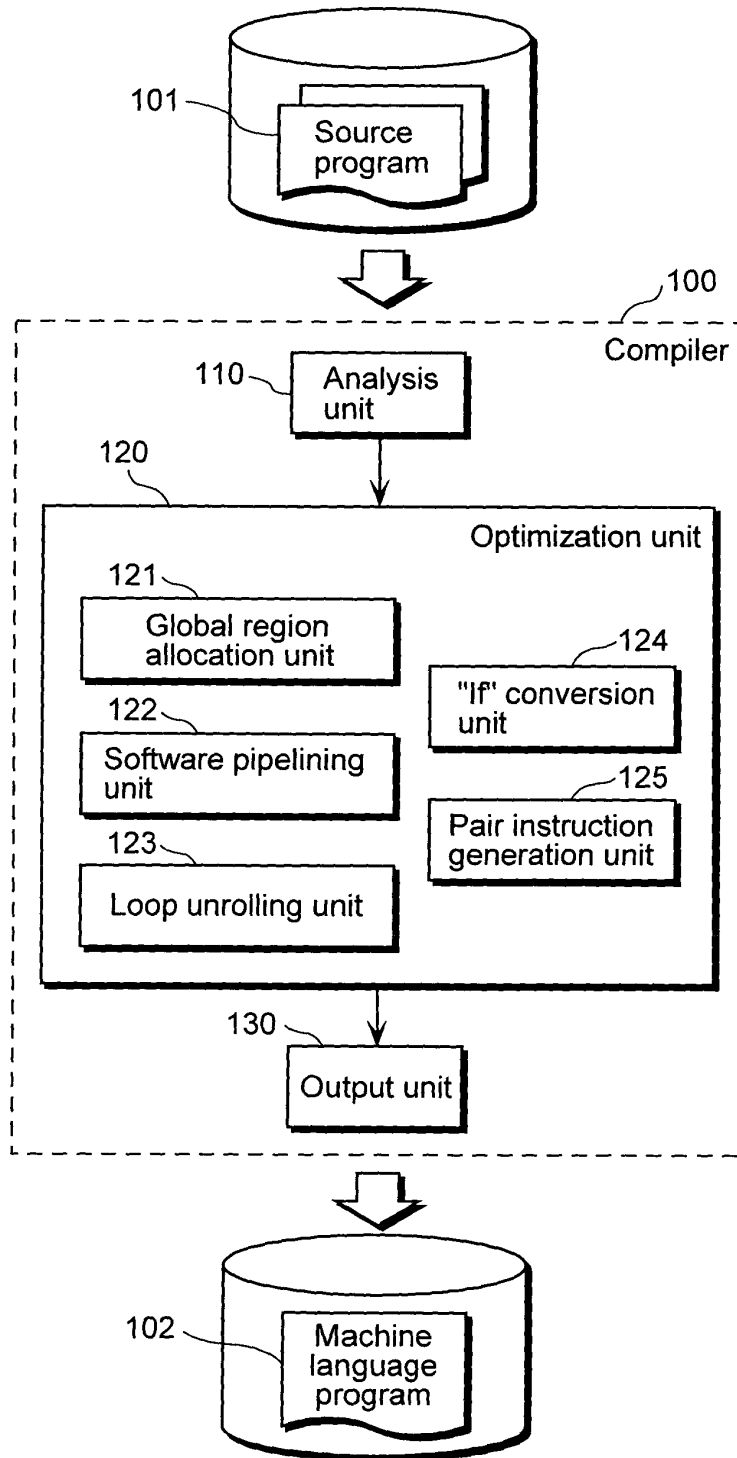


Fig. 38A

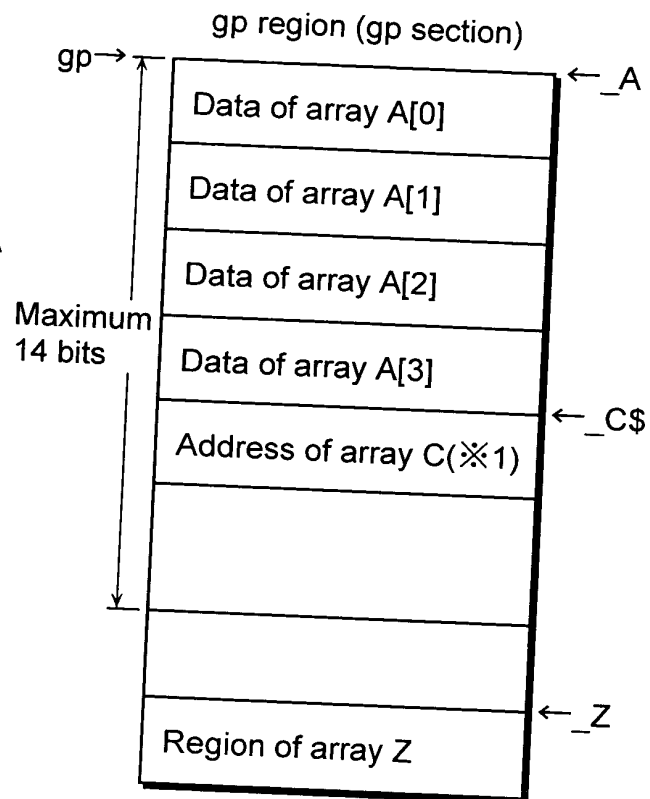


Fig. 38B

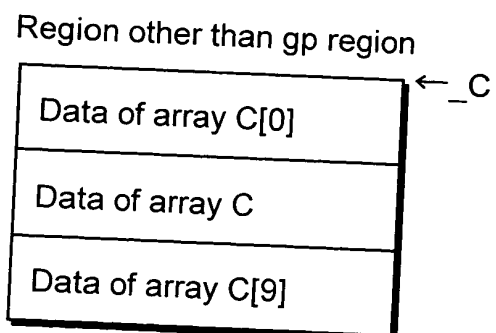


Fig. 39

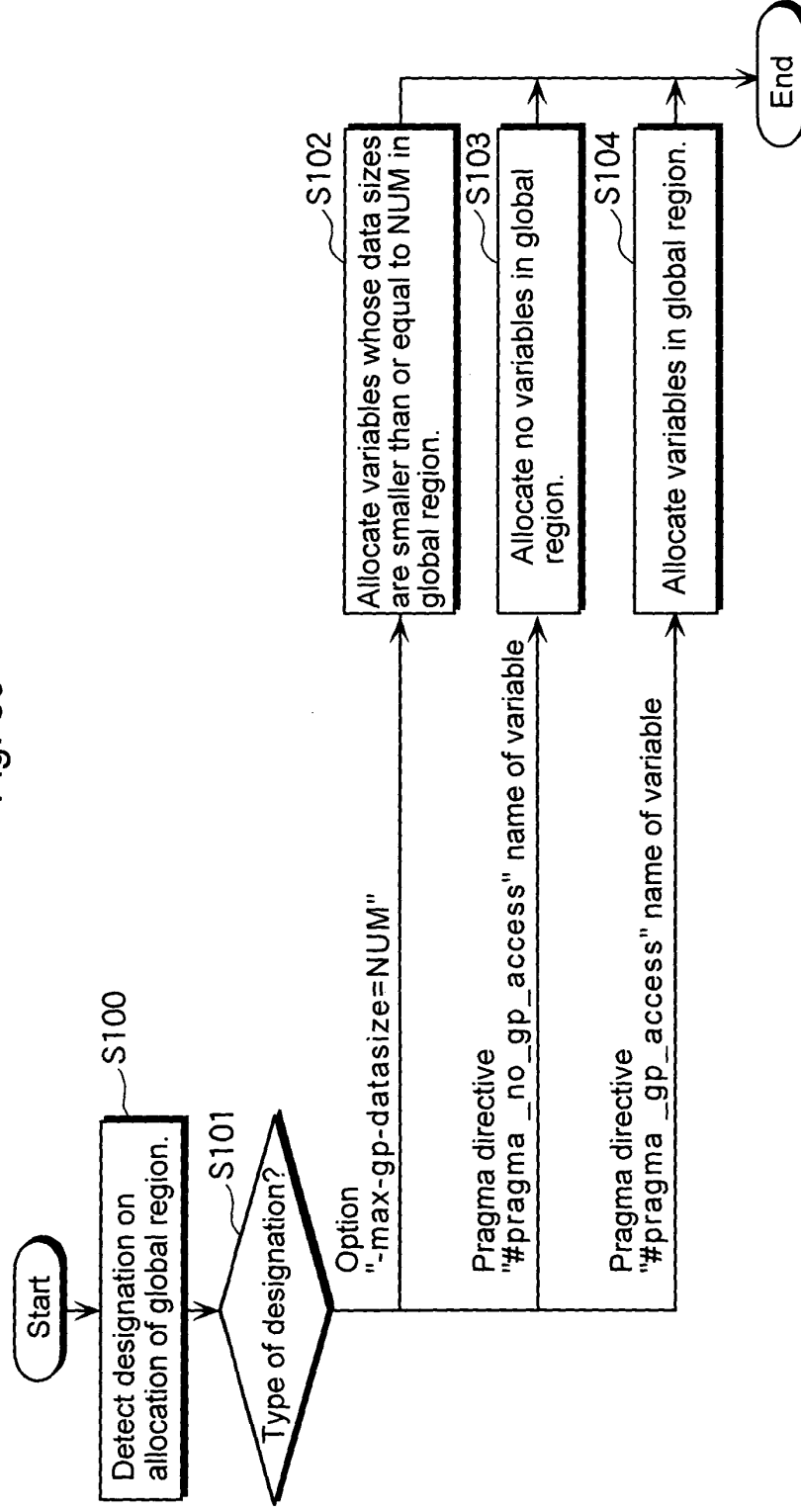


Fig. 40

Comparison of generated codes by change of maximum data size

Default	Designate "-mmax-gp-datasize=40"
<pre>int a[8]; int c[10]; int sample(void) { a[7] = c[0] + c[9]; return a[7]; }</pre>	<pre>int a[8]; int c[10]; int sample(void) { a[7] = c[0] + c[9]; return a[7]; }</pre>
<pre>ld r1,(gp,_c\$ - .MN.gptop);; setlo r0,LO(_c+36);; sethi r0,HI(_c+36);; ld r1,(r1);; ld r0,(r0);; add r0,r1,r0;; st (gp,_a - .MN.gptop+28),r0 ret;;</pre>	<pre>ld r1,(gp,_c - .MN.gptop);; ld r0,(gp,_c - .MN.gptop+36);; add r0,r1,r0;; st (gp,_a - .MN.gptop+28),r0 ret;;</pre>
<p>10 cycles 8 bytes</p>	<p>7 cycles 5 bytes</p>

Fig. 41

When maximum data size to be allocated to gp region is 40

No size designation Defined outside of file	Defined inside of file/Size designation Defined outside of file
<pre>extern int a[]; extern int c[]; int sample(void) { a[7] = c[0] + c[9]; return a[7]; }</pre>	<pre>int a[8]; extern int c[10]; int sample(void) { a[7] = c[0] + c[9]; return a[7]; }</pre>
<pre>setlo r0,LO(_c);; setlo r1,LO(_c+36) sethi r0,HI(_c);; sethi r1,HI(_c+36);; ld r3,(r0);; setlo r2,LO(_a+28) ld r0,(r1);; sethi r2,HI(_a+28);; add r1,r3,r0;; mov r0,r1 st (r2),r1 ret ;;</pre>	<pre>ld r1,(gp,_c - .MN.gptop);; ld r0,(gp,_c - .MN.gptop+36);; add r0,r1,r0;; st (gp,_a - .MN.gptop+28),r0 ret;;</pre>
<p>10 cycles 12 bytes</p>	<p>7 cycles 5 bytes</p>

Fig. 42

When maximum data size to be allocated to gp region is default of 32

Size designation Defined outside of file/ Defined inside of file "#pragma _no_gp_access" directive	No size designation Defined outside of file/ Defined inside of file "#pragma _gp_access" directive
<pre>#pragma _no_gp_access a, c extern int a[8]; int c[10]; int sample(void) { a[7] = c[0] + c[9]; return a[7]; }</pre>	<pre>#pragma _gp_access a, c extern int a[]; int c[10]; int sample(void) { a[7] = c[0] + c[9]; return a[7]; }</pre>
<pre>setlo r0,LO(_c);; setlo r1,LO(_c+36) sethi r0,HI(_c);; sethi r1,HI(_c+36);; ld r3,(r0);; setlo r2,LO(_a+28) ld r0,(r1);; sethi r2,HI(_a+28);; add r1,r3,r0;; mov r0,r1 st (r2),r1 ret ;;</pre>	<pre>ld r1,(gp,_c - .MN.gptop);; ld r0,(gp,_c - .MN.gptop+36);; add r0,r1,r0;; st (gp,_a - .MN.gptop+28),r0 ret;;</pre>
<p>10 cycles 12 bytes</p>	<p>7 cycles 5 bytes</p>

Fig. 43A

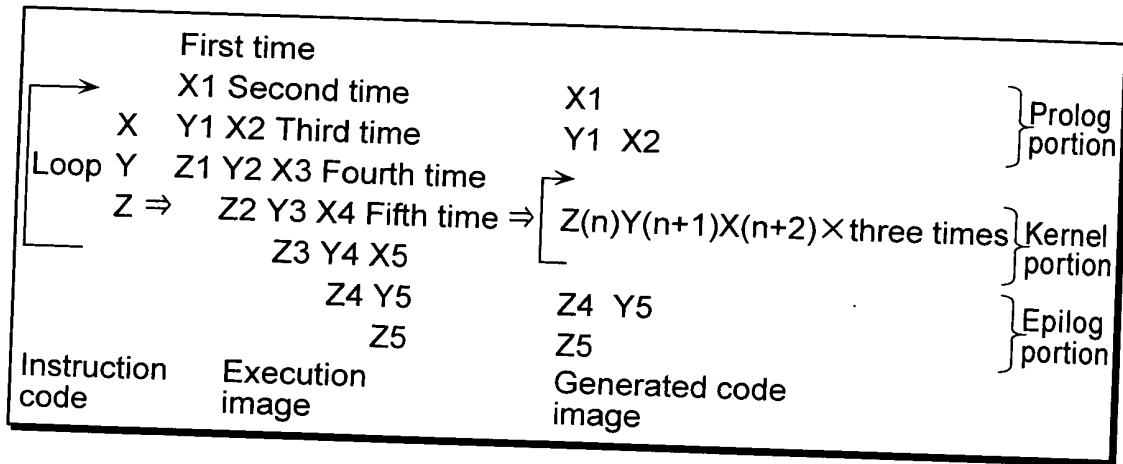


Fig. 43B

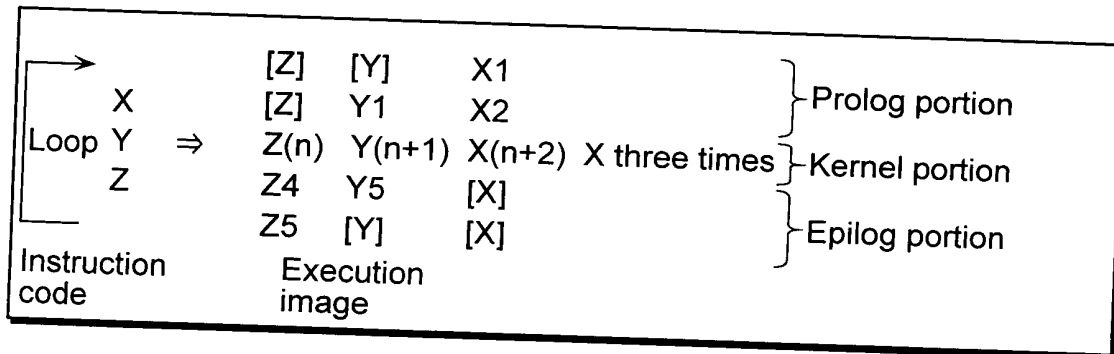
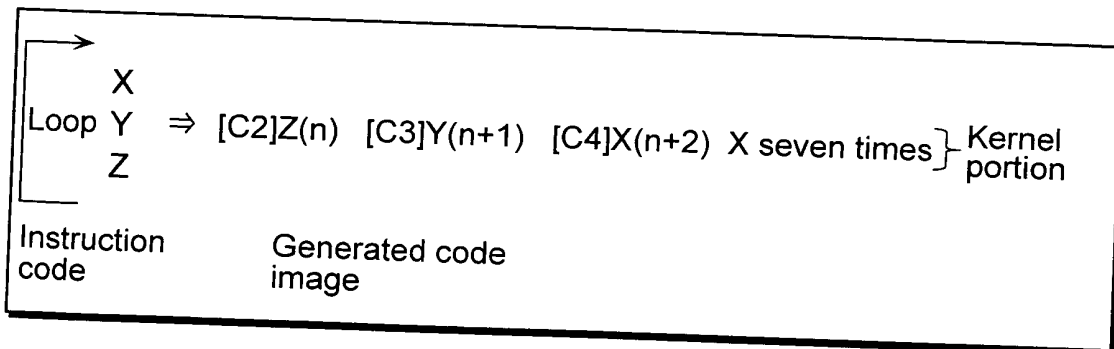


Fig. 43C



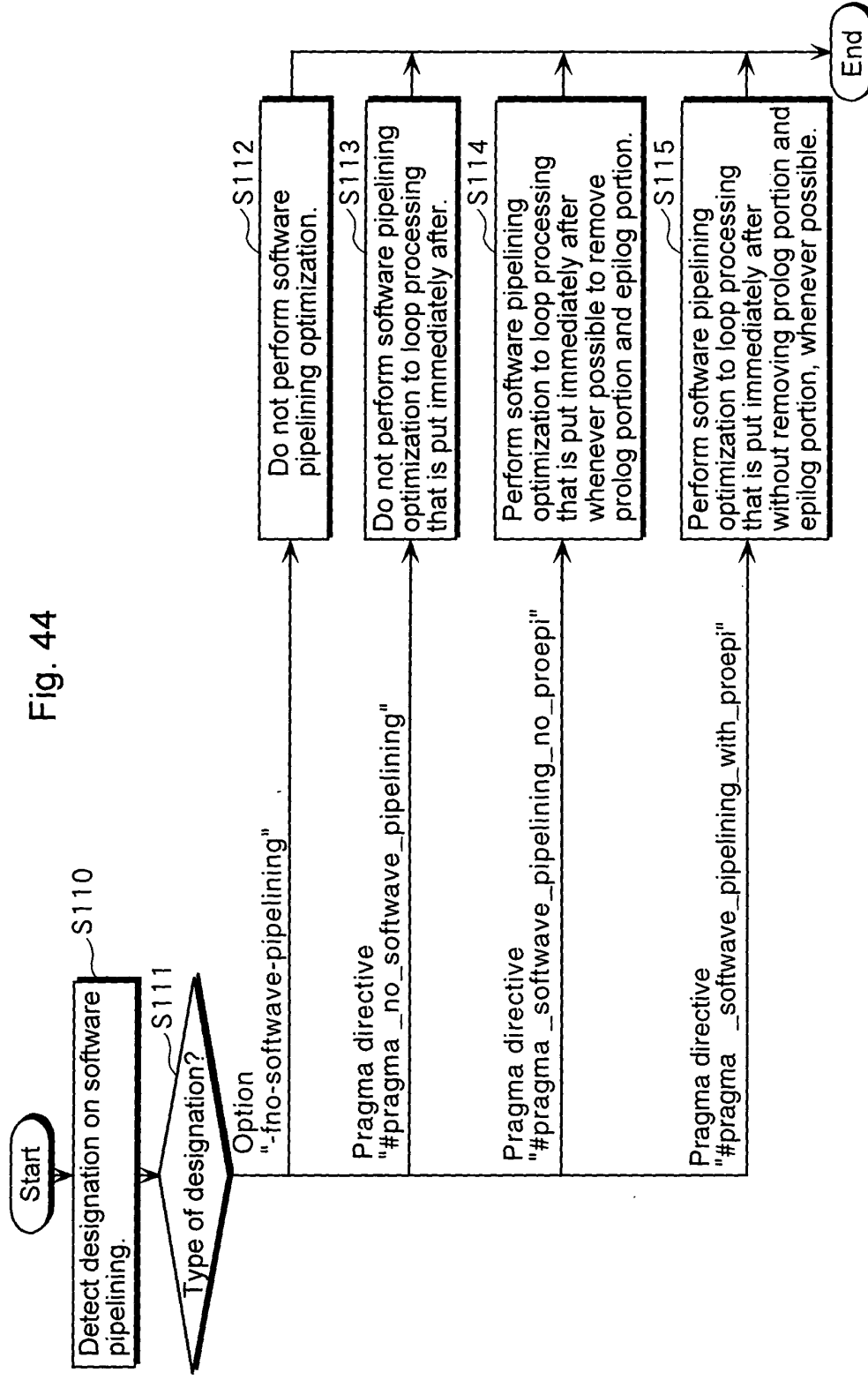


Fig. 45

Compilation with compile option -O

Default	"#pragma _software_pipelining_with_proepi" directive
<pre> int a[100]; int b[100]; void sample(int c) { int i; for(i=0; i<100; i++) { a[i] = b[i] + c; } } </pre>	<pre> int a[100]; int b[100]; void sample(int c) { int i; #pragma _software_pipelining_with_proepi for(i=0; i<100; i++) { a[i] = b[i] + c; } } </pre>
<pre> ld r5,(gp,_x\$ - .MN.gptop);; mov r1,98 settar C6,C4,L00030 ld r4,(gp,_v\$ - .MN.gptop);; L00030 [C4] add r2,r3,r0 [C6] ld r3,(r5+);; [C4] st (r4+),r2 [C6] jloop C6,C4,tar,r1,-1;; ret ;; </pre>	<pre> ld r5,(gp,_x\$ - .MN.gptop);; ld r4,(r5+);; mov r1,96 settar C6,L00023 ld r3,(gp,_v\$ - .MN.gptop);; L00023 add r2,r4,r0 ld r4,(r5+);; st (r3+),r2 [C6] jloop C6,C4,tar,r1,-1;; add r2,r4,r0;; st (r3+),r2 ret ;; </pre>
<p>2 + 2 × 101 + 3 = 207 cycles 9 bytes</p>	<p>3 + 2 × 99 + 3 = 204 cycles 12 bytes</p>

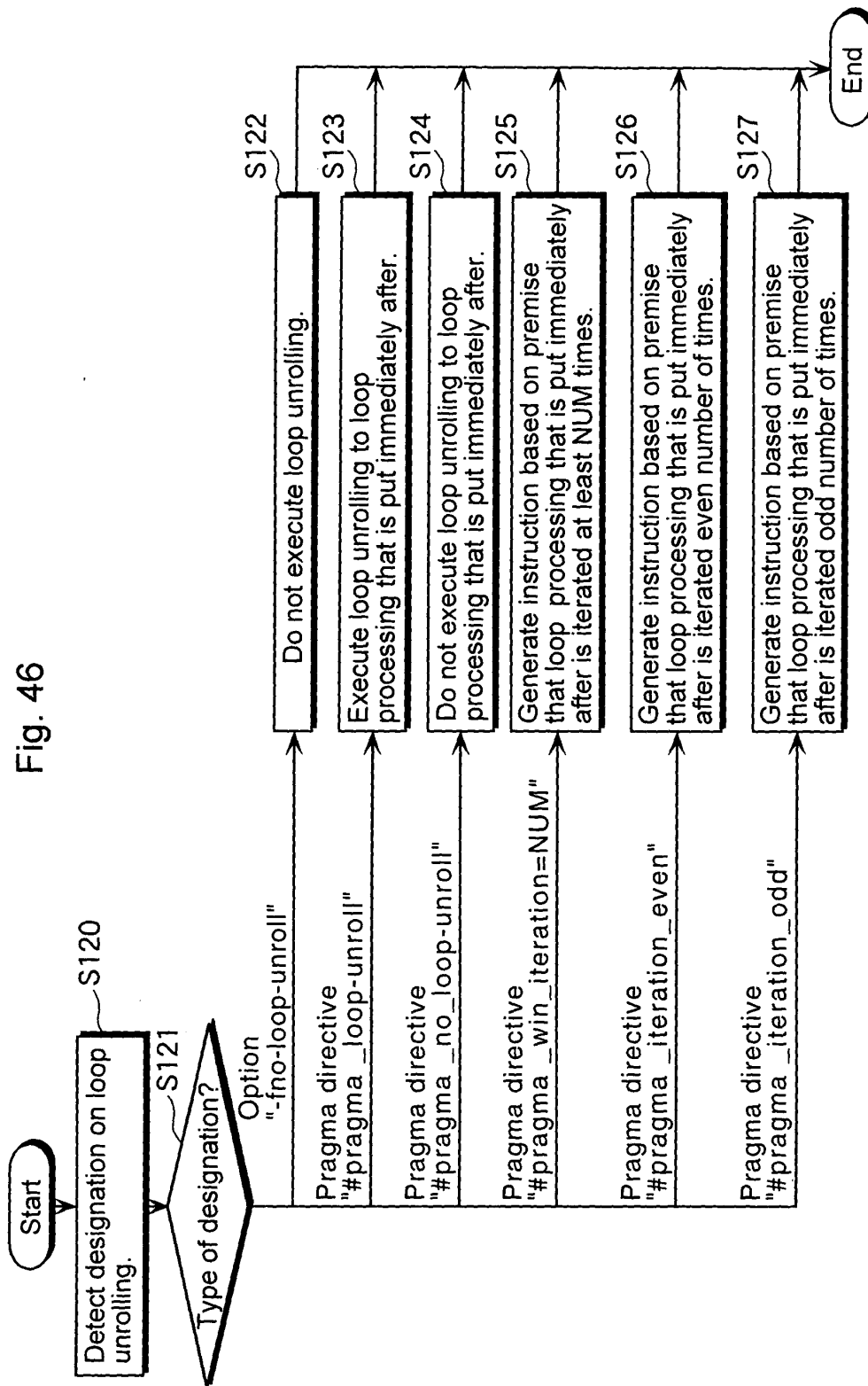


Fig. 47

Example of increased execution speed by "#pragma _loop_unroll" directive

Default	"#pragma _loop_unroll" directive
<pre> int a[100]; int b[100]; void sample(int c) { int i, ret=0; for(i=0; i<100; i++) { ret += a[i] * a[i]; } return ret; } </pre>	<pre> int a[100]; int b[100]; void sample(int c) { int i, ret=0; #pragma _loop_unroll for(i=0; i<100; i++) { ret += a[i] * a[i]; } return ret; } </pre>
<pre> mov r0,0 mov r1,98;; settar C6,C4,L00013 ld r3,(gp,_a\$ - .MN.gptop) mul m0,r2,gp,0;; L00013 [C4] mac m0,r0,r2,r2,m0 [C6] ld r2,(r3+) [C6] jloop C6,C4,tar,r1,-1;; ret ;; </pre>	<pre> ld r3,(gp,_a\$ - .MN.gptop);; mov r0,0 mov r1,48;; settar C6,C2:C4,L00013 add r5,r3,4 mul m0,r2,gp,0;; L00013 [C2] mac m0,r0,r2,r2,m0 [C3] add r3,r3,8 [C3] ld r2,(r5+)8;; [C3] mac m0,r0,r4,r4,m0 [C4] ld r4,(r3) [C6] jloop C6,C2:C4,tar,r1,-1;; ret ;; </pre>
<p>2 + 2 × 101 + 3 = 207 cycles 9 bytes</p>	<p>3 + 2 × 52 + 3 = 110 cycles 13 bytes</p>

Fig. 48

Default	"#pragma _loop_unroll" directive and "#pragma _align_local_pointer" directive
<pre> int a[100]; int b[100]; void sample(int c) { int i; int *pa=a; int *pb=b; for(i=0; i<100; i++) { *pa++ = i * c >> *pb++; } } </pre>	<pre> int a[100]; int b[100]; void sample(int c) { int i; #pragma _align_local_pointer=8 pa,pb int *pa=a; int *pb=b; #pragma _loop_unroll for(i=0; i<100; i++) { *pa++ = i * c >> *pb++; } } </pre>
<pre> mov r4,0 ld r6,(gp,_a\$ - .MN.gptop);; mov r1,98 settar C6,C4,L00025 ld r5,(gp,_b\$ - .MN.gptop);; L00025 [C4] asr r2,r4,r3;; [C6] ld r3,(r5+);; [C4] add r4,r4,r0 [C4] st (r6+),r2 [C6] jloop C6,C4,tar,r1,-1;; ret ;; </pre>	<pre> mov r6,0 ld r8,(gp,_a\$ - .MN.gptop);; mov r1,48 settar C6,C4,L00016 ld r7,(gp,_b\$ - .MN.gptop);; L00016 [C2] add r6,r6,r0;; [C2] stp (r8+),r2:r3 [C3] asr r2,r6,r4 [C3] add r6,r6,r0;; [C3] asr r3,r6,r5 [C4] ldp r4:r5,(r7+) [C6] jloop C6,C2:C4,tar,r1,-1;; ret ;; </pre>
<p>2 + 3 × 101 + 3 = 308 cycles 11 bytes</p>	<p>2 + 3 × 51 + 3 = 158 cycles 13 bytes</p>

Fig. 49

Without "#pragma _min_iteration" directive	With "#pragma _min_iteration" directive
<pre> int a[101]; int b[101]; void sample(int c, int end) { int i; int *pa=a; int *pb=b; for (i = 0; i <end; i++) { *pa++ = i * c >> *pb++; } *pa = end; } </pre>	<pre> int a[101]; int b[101]; void sample(int c, int end) { int i; int *pa=a; int *pb=b; #pragma _min_iteration=4 for (i = 0; i <end; i++) { *pa++ = i * c >> *pb++; } *pa = end; } </pre>
<pre> cmple C0,r1,0 ld r5,(gp,_a\$ - .MN.gptop);; mov r4,0 ld r6,(gp,_b\$ - .MN.gptop) [C0] br L00016;; mov r3,0 settar L00017 L00017 ld r2,(r6+);; add r4,1;; cmplt C0,r4,r1 asr r2,r3,r2;; add r3,r3,r0 st (r5+),r2 [C0] jmpf tar;; L00016 st (r5),r1 ret ;; </pre>	<pre> mov r5,0 ld r7,(gp,_a\$ - .MN.gptop);; settar C6,C4,L00027 sub r2,r1,2 ld r6,(gp,_b\$ - .MN.gptop);; L00027 [C4] asr r3,r5,r4;; [C6] ld r4,(r6+);; [C4] add r5,r5,r0 [C4] st (r7+),r3 [C6] jloop C6,C4,tar,r2,-1;; st (r7),r1 ret ;; </pre>
<p>2 + 4 × 100 + 3 = 405 cycles 16 bytes</p>	<p>2 + 3 × 101 + 3 = 308 cycles 12 bytes</p>

Fig. 50

Example of loop unrolling being impossible

C language source with unknown number of loops	Result of compilation
<pre> void sample(int c, int end) { int i; #pragma _align_local_pointer=8 pa, pb; int *pa=a; int *pb=b; #pragma _loop_unroll #pragma _min_iteration=4 for (i = 0; i <end; i++) { *pa++ = i * c >> *pb++; } *pa = end; } </pre>	<pre> mov r5,0 ld r7,(gp,_a\$ - .MN.gptop);; settar C6,C4,L00026 sub r2,r1,2 ld r6,(gp,_b\$ - .MN.gptop);; L00026 [C4] asr r3,r5,r4;; [C6] ld r4,(r6+);; [C4] add r5,r5,r0 [C4] st (r7+),r3 [C6] jloop C6,C4,tar,r2,-1;; st (r7),r1 ret ;; </pre>

Fig. 51

"#pragma _iteration_even" directive	"#pragma _iteration_odd" directive
<pre> int a[101]; int b[101]; void sample(int c, int end) { int i; #pragma _align_local_pointer=8 pa, pb int *pa=a; int *pb=b; #pragma _min_iteration=50 #pragma _loop_unroll #pragma _iteration_even for (i = 0; i <end; i++) { *pa++ = i * c >> *pb++; } *pa = end; } </pre>	<pre> int a[101]; int b[101]; void sample(int c, int end) { int i; #pragma _align_local_pointer=8 pa, pb int *pa=a; int *pb=b; #pragma _min_iteration=50 #pragma _loop_unroll #pragma _iteration_odd for (i = 0; i <end; i++) { *pa++ = i * c >> *pb++; } *pa = end; } </pre>
<pre> sub r2,r1,6;; mov r6,0 asr r2,1 ld r7,(gp,_a\$ - .MN.gptop);; settar C6,C4,L00036 add r2,1 ld r8,(gp,_b\$ - .MN.gptop);; L00036 [C4] asr r3,r6,r4;; [C4] add r6,r6,r0 [C4] st (r7+),r3;; [C4] asr r3,r6,r5;; [C6] ldp r4:r5,(r8+);; [C4] add r6,r6,r0 [C4] st (r7+),r3 [C6] jloop C6,C4,tar,r2,-1;; st (r7),r1 ret ;; </pre>	<pre> sub r2,r1,7;; mov r6,0 asr r2,1 ld r7,(gp,_a\$ - .MN.gptop);; settar C6,C4,L00046 add r2,1 ld r8,(gp,_b\$ - .MN.gptop);; L00046 [C4] asr r3,r6,r4;; [C4] add r6,r6,r0 [C4] st (r7+),r3;; [C4] asr r3,r6,r5;; [C6] ldp r4:r5,(r8+);; [C4] add r6,r6,r0 [C4] st (r7+),r3 [C6] jloop C6,C4,tar,r2,-1;; ld r2,(r8+);; asr r2,r6,r2;; st (r7+),r2;; st (r7),r1 ret ;; </pre>

Fig. 52

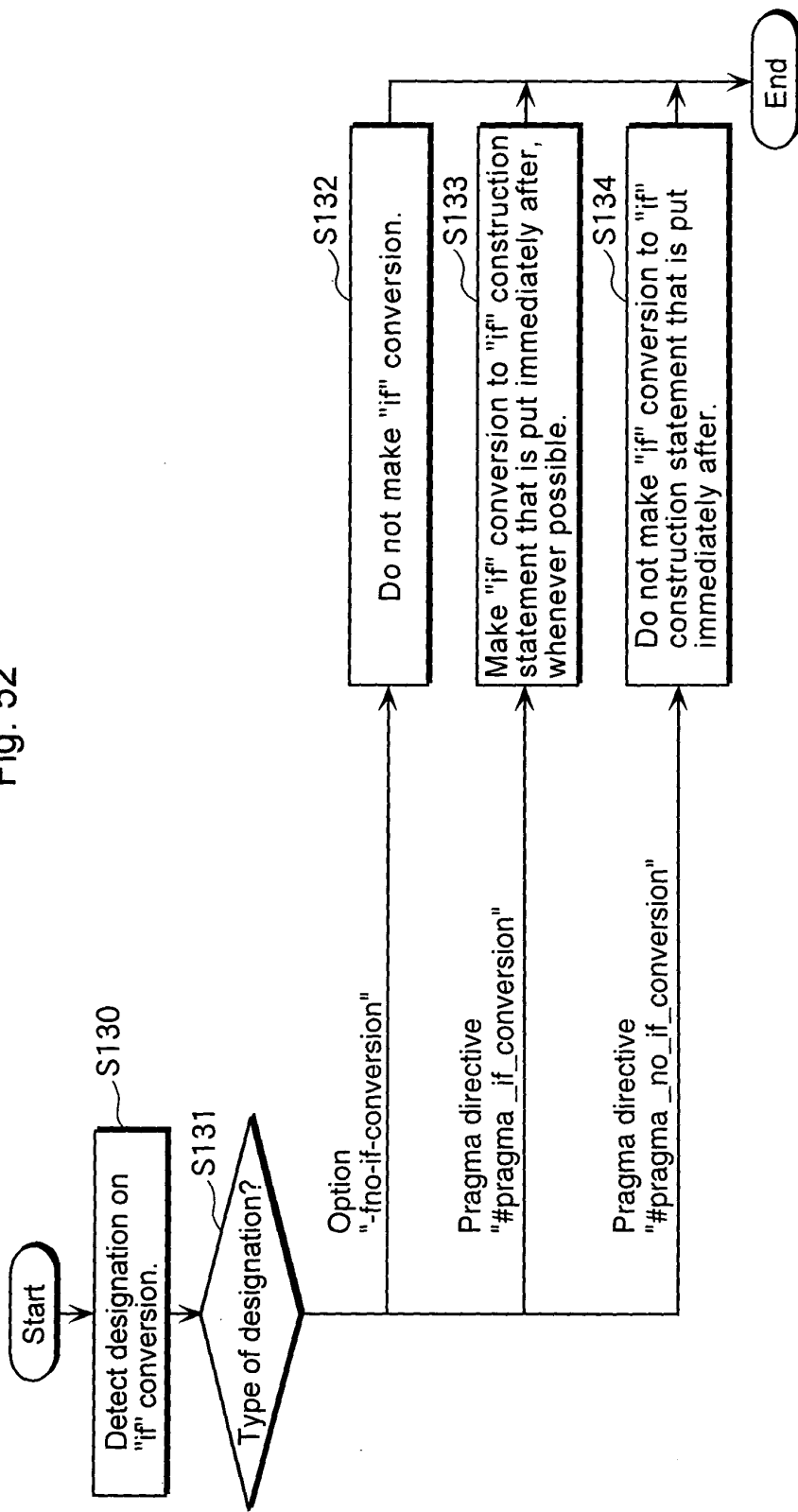


Fig. 53

"#pragma_no_if_conversion" directive	"#pragma_if_conversion" directive
<pre> int func(int a) { int b ; #pragma_no_if_conversion if(a == 2) b = 0 ; else b = 1 ; return b ; } </pre>	<pre> int func(int a) { int b ; #pragma_if_conversion if(a == 2) b = 0 ; else b = 1 ; return b ; } </pre>
<pre> cmpne C0,r0,2 ;; [C0] br L00002 ;; // Branch is generated. mov r0,0 ret ;; L00002 mov r0,1 ret ;; </pre>	<pre> cmpeq C0:C1,r0,2 ;; [C0] mov r0,0 // Execution instruction with condition is [C1] mov r0,1 // generated instead of branch instruction. ret ;; </pre>
"then" side: 5 cycles, "else" side 7 cycles 12 bytes	4 cycles 8 bytes

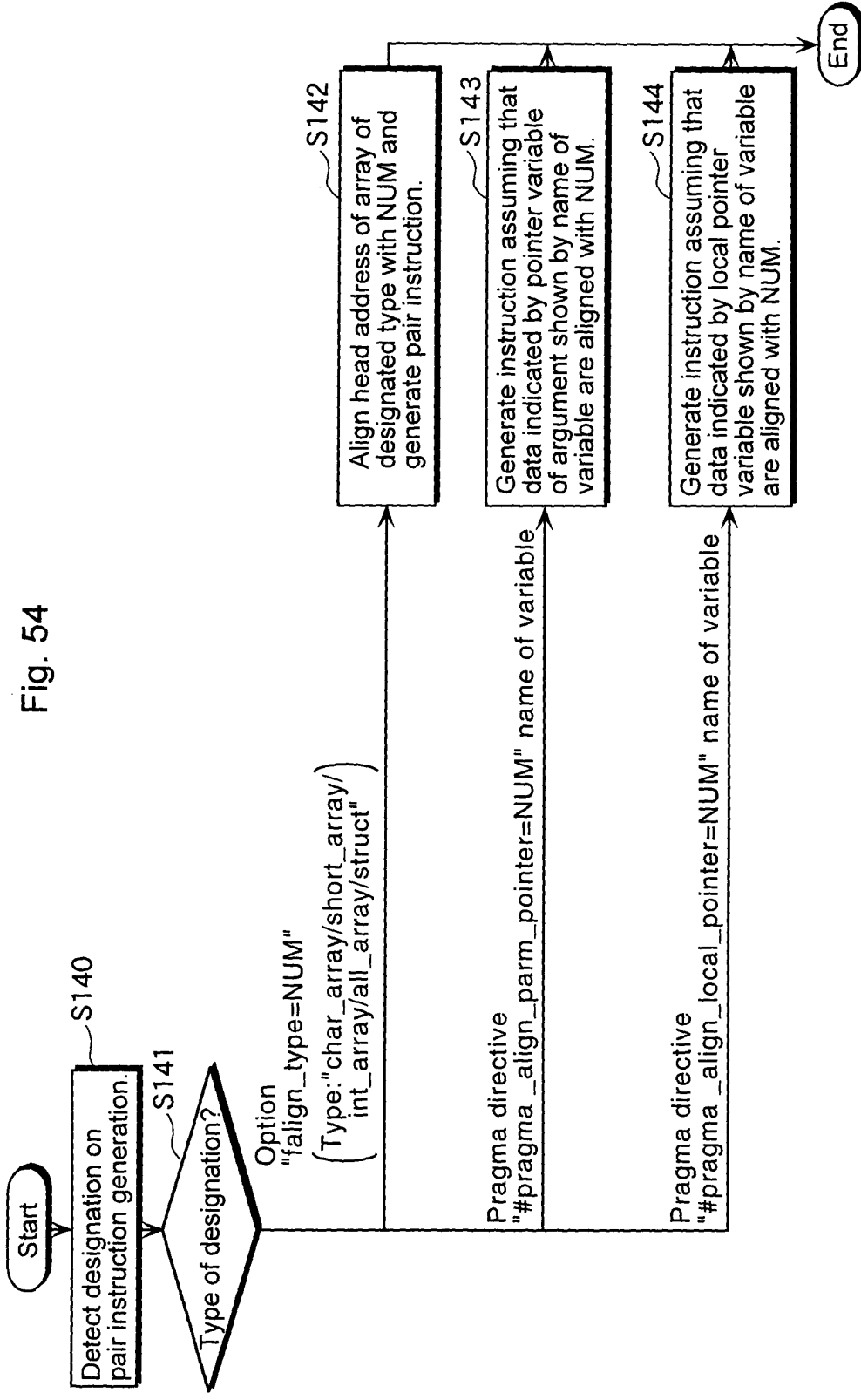


Fig. 54

Fig. 55

No designation of alignment	Designation of option "-falign_short_array=4"
<pre> short src[8]; short dst[8]; void align1(void) { char dummy; int i; for(i=0; i<8; i+=2) dst[i] = src[i] + src[i+1]; } </pre>	<p>No change</p>
<pre> addu r5, gp, _src - .MN.gptop addu r4, gp, _src - .MN.gptop+2 ;; addu r3, gp, _dst - .MN.gptop mov r0, 2 settar C6, C4, L00005 ;; L00005 [C4] ldh r1, (r4)+4 ;; // As alignment is unknown, [C4] add r1, r2, r1 // two data are loaded [C6] ldh r2, (r5)+4 ;; // independently (ldh instruction) [C4] sth (r3)+4, r1 [C6] jloop C6, C4, tar, r0, -1 ;; //L00005 ret ;; </pre>	<pre> addu r5, gp, _src - .MN.gptop ;; addu r4, gp, _dst - .MN.gptop mov r0, 2 settar C6, C4, L00005 ;; L00005 [C4] add r1, r2, r3 // As alignment is 4, [C6] ldhp r2: r3, (r5+) ;; // pair access instruction [C4] sth (r4)+4, r1 // (ldhp) is generated. [C6] jloop C6, C4, tar, r0, -1 ;; //L00005 ret ;; </pre>
<p>2+4×5+3=25 cycles 22 bytes</p>	<p>2+2×5+3=15 cycles 18 bytes</p>

Fig. 56

No designation of alignment	Use "#pragma _align_ parm_pointer" directive
<pre> int align2(short * src) { int v = 0 ; for(int i=0 ; i<100 ; i+=2){ v += src[i] * src[i+1] ; } return v ; } </pre>	<pre> #pragma _align_parm_pointer=4 src int align2(short * src) { int v = 0 ; for(int i=0 ; i<100 ; i+=2){ v += src[i] * src[i+1] ; } return v ; } </pre>
<pre> mov r5,0 mov r1,48 settar C6,C4,L00005 ;; mov r4,r0 add r3,r0,2 mul m0,r2,gp,0 ;; L00005 [C4] ldh r0,(r3+)4 ;; // Apply software pipelining. [C4] lmac m0,r5,r2,r0,m0 // As alignment of data is unknown, [C6] ldh r2,(r4+)4 // execute ldh instruction twice. [C6] jloop C6,C4,tar,r1,-1 ;; //L00005 mov r0,r5 ret ;; </pre>	<pre> mov r4,0 ;; mov r1,48 settar C6,C4,L00005 mul m0,r2,gp,0 ;; L00005 [C4] lmac m0,r4,r2,r3,m0 // Apply software pipelining. [C6] ldhp r2:r3,(r0+) // Pair access instruction is usable. [C6] jloop C6,C4,tar,r1,-1 ;; //L00005 mov r0,r4 ret ;; </pre>
<p>2+3×51+3=160 cycles 24 bytes</p>	<p>2+2×51+3=107 cycles 18 bytes</p>

Fig. 57

No designation of alignment	Use "#pragma _align_local_pointer" directive
<pre> void align3(int n) { short * from ; short * to ; int i ; from = &(src[n]) ; to = &(dst[n]) ; for(i=0 ; i<16 ; i++, from+=2, to+=2){ * to = * from ; *(to+1) = * (from+1) ; } } </pre>	<pre> void align3(int n) { #pragma _align_local_pointer=4 from,to short * from ; short * to ; int i ; from = &(src[n]) ; to = &(dst[n]) ; for(i=0 ; i<16 ; i++, from+=2, to+=2){ * to = * from ; *(to+1) = * (from+1) ; } } </pre>
<pre> ld r3,(gp,_src\$ - .MN.gptop) ;; ld r2,(gp,_dst\$ - .MN.gptop) ;; add r1,r0,r0 ;; mov r0,13 add r6,r3,r1 add r5,r2,r1 ;; settarC6,L00016 add r4,r5,2 add r3,r6,2 ;; L00016 ldh r1,(r6+)4 ;; ldh r2,(r3+)4 ;; sth (r5+)4,r1 ;; sth (r4+)4,r2 [C6] jloop C6,tar,r0,-1 ;; //L00016 ret ;; </pre>	<pre> ld r3,(gp,_src\$ - .MN.gptop) ;; ld r2,(gp,_dst\$ - .MN.gptop) ;; mov r1,13 add r0,r0,r0 ;; settarC6,L00016 add r4,r3,r0 add r0,r2,r0 ;; L00016 ldhp r2:r3,(r4+) ;; sthp (r0+),r2:r3 [C6] jloop C6,tar,r1,-1 ;; //L00016 ret ;; </pre>
<p>5+4 × 16+3=72 cycles 30 bytes</p>	<p>5+3 × 16+3=56 cycles 22 bytes</p>